UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

ESTUDIO DEL PROBLEMA DE LA DISTRIBUCIÓN DE CLAVES CRIPTOGRÁFICAS EN EL CONTEXTO DE LAS TECNOLOGÍAS *CLOUD*

(COSITI_128/1314)

Alejandro Sánchez Gómez

Julio 2014

ESTUDIO DEL PROBLEMA DE LA DISTRIBUCIÓN DE CLAVES CRIPTOGRÁFICAS EN EL CONTEXTO DE LAS TECNOLOGÍAS *CLOUD*

(COSITI_128/1314)

AUTOR: Alejandro Sánchez Gómez

TUTOR: David Arroyo Guardeño

PONENTE: Luis Fernando Lago Fernández

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Julio 2014

Agradecimientos

Me gustaría agradecer en estas líneas a todas aquellas personas que me han ayudado a lograr todos los objetivos que me he propuesto y llegar a finalizar este proyecto.

A mi tutor David por haberme ayudado con el Trabajo de Fin de Grado, porque siempre ha estado apoyándome y guiándome en todo momento que lo he necesitado.

A todos los profesores que he tenido a lo largo de estos cuatro años, siendo siempre muy cercanos al alumno y preocupándose mucho por su aprendizaje.

A mis amigos de toda la vida, que siempre te ayudan y te sacan una sonrisa cuando más lo necesitas.

A mis compañeros que he conocido en la universidad, en especial a Carlos, Borja e Iván que han hecho que estos cuatro años sean más llevaderos.
A Laura, por haberme ayudado siempre que se lo he pedido cuando compartíamos asignaturas. A Rus, por haberme echado una mano en todos los obstáculos que han aparecido a lo largo de la carrera y haber encontrado una gran amiga.

Y por supuesto, a mi familia, que es lo más importante y gracias a su esfuerzo he podido realizar estos cuatro años de estudios.

A todos ellos, muchísimas gracias.

Resumen

Uno de los principales problemas de la criptografía contemporánea reside en la generación y distribución de claves criptográficas. En el contexto de las tecnologías *cloud* este problema se agudiza. En efecto, el propietario de los activos delega la custodia de los mismos en una tercera parte, en principio, no confiable. De esta forma, el único recurso del que dispone un usuario medio para evitar que un servidor pueda acceder a sus activos de información es el cifrado de los mismos. Así, un usuario podría hacer uso de una aplicación cliente encargada de cifrar la información antes de enviarla al servidor *cloud* con el que habitualmente trabaja. En el caso de que el usuario pertenezca a un grupo de trabajo, deberá compartir la clave de cifrado para que los componentes de ese grupo puedan acceder al recurso.

El objetivo principal del presente trabajo es poder estudiar e ilustrar la dificultad existente en la distribución de claves criptográficas simétricas en entornos *cloud*. Una vez efectuado el pertinente análisis del estado del arte, a modo de prueba de concepto, se propondrá un prototipo de cliente capaz de poder realizar el cifrado de todos los activos que un usuario quiera subir a un servidor *cloud*, además de poder descargarlos y compartirlos. Este enfoque proporciona la seguridad de la cual no dispone el usuario al depositar su información directamente en estos tipos de entorno. El cliente desarrollado garantiza la confidencialidad y la integridad de los datos, y además es el encargado de realizar todas las operaciones criptográficas, dejando al servidor fuera de estas tareas. Asimismo, el servidor *cloud* asegura la disponibilidad de los recursos y un primer nivel de autenticación de un usuario, ya que se encarga de comprobar sus credenciales de acceso al servidor.

El cliente que se va a presentar ha sido desarrollado sobre la plataforma *Android*, debido a su gran cuota de mercado actual, y la disponibilidad de una completa plataforma de desarrollo. Por otro lado, se ha elegido *Dropbox* como servidor *cloud*, porque es bastante conocido tanto a nivel de usuarios como de empresas, y proporciona una API de desarrollo que es fácil de usar.

Finalmente destacar que se ha optado por una implementación *OpenSource*, debido a las numerosas ventajas que ofrece este enfoque. Este tipo de implementación tiene una mayor fiabilidad, al ser testeados, usados y corregidos en diferentes entornos. Además, aporta una gran flexibilidad al usuario, pudiendo estudiar cómo funciona el código, para poder modificarlo y adaptarlo a unos requerimientos particulares. Por último, se da libertad de mejorar la implementación realizada para poner esas mejoras a disposición del público, con el fin de que toda la comunidad se beneficie.

Palabras clave

OpenSource, *Android*, Seguridad, *Dropbox*, *Cloud*, Distribución de claves, Criptografía, AES 256, RSA, Firma digital.

Abstract

One of the main problems of contemporary cryptography lies in the generation and distribution of cryptographic keys. In the context of cloud technologies, this problem becomes even more critical. Indeed, the asset owner delegates the custody of her assets to a third party, which is not reliable in principle. Information encryption is the only mean for an average user to prevent a server accessing her information assets. Thus, a user could use a client application for data encryption before sending it to her usual cloud server. In the case that the user belongs to a workgroup, resource sharing requires the user sending the encryption key to the rest of components of the group.

The main objective of this work is to study and illustrate the difficulty in distributing symmetric cryptographic keys in cloud environments. Upon completion of the relevant state of the art analysis, and as a proof of concept, a prototype client is proposed to encrypt all assets before a user performs its uploading to a cloud server. This client also enables the download and share of the assets managed through the cloud server. This approach provides the security that the user does not have to place her information directly into these types of environment. The client developed ensures the data confidentiality and integrity, and is also be responsible for performing all cryptographic operations, leaving the server out of these tasks. Finally, the cloud server guarantees resources availability and a first level of user authentication, which consists of validating users' credentials to accessing the server.

The application client of this project has been developed on the *Android* platform, due to its high current market share, and the availability of a complete development platform. On the other hand, it has been chosen *Dropbox* as a cloud server, because it is well known by both users and companies, and provides a development API that is easy to use.

Finally note that it has been chosen an *OpenSource* implementation, since it has many advantages. This type of deployment is more reliable because the so-implemented products are tested, used and fixed in different environments by many different users and programmers. They also provide great flexibility to the user, because you can study how the code works, so you can modify and adapt to particular requirements. In short, this approach gives freedom to improve the implementation and to make it public, so that the whole community benefits from it.

Key Words

OpenSource, *Android*, Security, *Dropbox*, *Cloud*, Key distribution, Cryptography, AES 256, RSA, Digital signature.

Índice de contenidos

Índice de tablas

Índice de figuras

Capítulo 1
Introducción

1.1. Objetivo

El desarrollo de las Tecnologías de la Información y de la Comunicación (TIC) es un componente esencial del actual modelo económico [1]. Dentro de las vigentes TIC está cobrando cada vez más interés el almacenamiento y tratamiento de información en la nube (*Cloud Computing)*. No es ninguna novedad que el paradigma *Cloud Computing* es una creciente tendencia en la computación, que cada día va teniendo un papel más protagónico en la vida de todas las personas. Sin embargo, se debe tener en cuenta que cuando un usuario deposita sus activos en un servidor de este tipo de tecnología, no hay garantías en la gestión de los mismos, ya que no se puede confiar plenamente en dicho servidor. Teniendo en cuenta la dependencia del modelo negocio de nuestra sociedad respecto a las TIC, debe evitarse cualquier práctica que erosione la confianza de los usuarios en dichas tecnologías [2]. Es por ello que en el presente trabajo se abordará la necesidad de dotar los *entornos cloud* con mecanismos seguros y usables para la preservación de los activos de información.

De modo general, y en el caso particular de las tecnologías *cloud*, un usuario puede proteger sus recursos mediante la criptografía. Así, si un usuario quiere subir un fichero a un servidor *cloud* evitando que éste acceda al recurso, previamente ha de cifrarlo. En el caso de que quiera compartir en un grupo de trabajo uno de esos activos subido a este tipo de servidores, tendrá que distribuir una clave criptográfica. Es importante tener en cuenta que en la criptografía actual, la generación y distribución de claves criptográficas es uno de los principales problemas existentes.

El objetivo de este proyecto es poder estudiar e ilustrar la dificultad que tiene la distribución de claves criptográficas en un entorno *cloud* ampliamente conocido. Para ello, se ha implementado un prototipo *OpenSource* llamado *EncryptedCloud*, capaz de subir, descargar y compartir cualquier tipo de archivo cifrado entre diferentes cuentas de *Dropbox*. Esta aplicación ha sido desarrollada para cualquier dispositivo móvil cuyo sistema operativo sea *Android*, a partir de la versión de *firmware* 2.3.6. Se trata de un modelo cliente-servidor que va a ser descrito detalladamente en el *Capítulo 4*.

1.2. Implementaciones similares existentes

En la actualidad, no se puede decir que haya muchas implementaciones existentes similares a la que se ha realizado, aunque sí que hay varias que se han dado a conocer, como pueden ser *Cloudfogger* [3] o *Safemonk* [4].

Sin embargo, el problema de la mayoría de estas implementaciones es que son soluciones propietarias. Para empezar, hay que registrarse en un servicio que almacena tus datos de autenticación. Por tanto, también disponen de un servidor en el cual tampoco se debe confiar si se quiere tener un control total sobre tus datos. Además, en

estos tipos de servicios la seguridad es más difícil de verificar que en los proyectos *OpenSource*, ya que en este último caso el código está disponible para el público general.

La implementación que se ha realizado es de código abierto. Toda la gestión criptográfica y de claves es realizada en el lado del cliente, por lo que el servidor no tiene ninguna sobrecarga criptográfica. Sólo de esta forma se le puede garantizar al usuario que tiene el total control sobre sus datos. *Dropbox* únicamente se utiliza para poder almacenar los datos, pero no para preservar la confidencialidad de éstos. El servidor de *Dropbox* podrá modificar o destruir los datos, pero los usuarios podrán detectar este comportamiento malicioso.

1.3. Estructura del documento

El presente documento tiene una estructura dividida en cinco capítulos distintos.

El primer capítulo es el actual, en el cual se describe el objetivo del trabajo realizado y se detallan otras implementaciones existentes similares a la desarrollada.

En el segundo capítulo se realiza una descripción de la computación en la nube. Se presentan sus características principales, las ventajas y desventajas que puede tener el uso de esta tecnología, y finalmente se enumeran los principales servicios de almacenamiento que son más populares en la actualidad.

En el tercer capítulo, se expone un breve estudio sobre la seguridad en los entornos *cloud*. La seguridad en estos entornos es un tema de gran importancia en los tiempos actuales, debido a la gran cantidad de usuarios que se están "subiendo a la nube".

En el cuarto capítulo se describe el proyecto software que se ha implementado. Se detallan aspectos relacionados con su planificación, su diseño, su implementación, su funcionalidad y un completo plan de pruebas.

Por último, en el capítulo quinto, se concluye el trabajo realizado y se proponen varias mejoras sobre el proyecto implementado que pueden servir como un trabajo futuro.

Capítulo 2
La computación en la nube

2.1. Introducción

La computación en la nube (*Cloud Computing*) es un paradigma que permite ofrecer servicios de computación a través de Internet (*Ver Figura 1*), de forma que los usuarios, sin necesidad de que tengan altos conocimientos informáticos, pueden acceder a distintos servicios desde cualquier dispositivo que sea compatible. La información se aloja en servidores remotos, accediendo a ella una vez que el usuario se ha autenticado con unas credenciales personales e intransferibles [5].

En la nube se debe diferenciar entre dos entidades completamente distintas. Por un lado se encuentra el Consumidor, que es el actor principal que utiliza los servicios de la computación en la nube. Por otro lado se encuentra el Proveedor, que es el encargado de proporcionar un servicio desde la nube a los Consumidores.

Este modelo permite aumentar el número de servicios basados en la red. Esto genera beneficios tanto para los Proveedores, que pueden ofrecer de forma más rápida y eficiente un mayor número de servicios, como para los usuarios, que pueden acceder a ellos de forma instantánea, disfrutando de un modelo de pago por consumo [6].

En definitiva, la computación en la nube sugiere un nuevo modelo de prestación de servicios y tecnología, por lo que cualquier persona o entidad interesada en innovar en su negocio debería considerar migrar sus servicios a los entornos *cloud*.

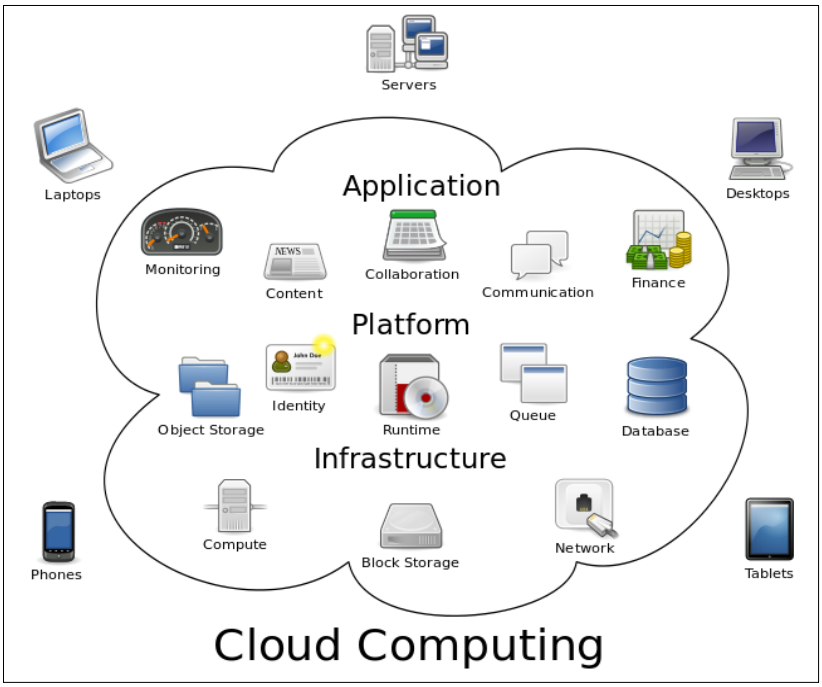


Figura 1. Esquema general de la computación en la nube

En este capítulo primero se va proporcionar una definición sobre la tecnología conocida como *Cloud Computing*, además de enumerar sus características más representativas. Se detallarán cuáles son algunas de las ventajas y desventajas de este tipo de computación. También se explicará su estructura, y se terminará facilitando un listado de cuatro de los principales servicios de almacenamiento en la nube existentes en la actualidad.

2.2. Definición

El *Instituto Nacional de Estándares y Tecnología* (NIST) definió el término *Cloud Computing* como un modelo que permite el acceso bajo demanda a un conjunto de recursos configurables (como redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados con un esfuerzo de gestión mínimo por parte de los Proveedores. Los Consumidores pueden utilizar estos recursos para desarrollar y ejecutar los servicios y aplicaciones según la demanda de una manera flexible, en cualquier dispositivo, momento y lugar [7].

2.3. Características principales

A medida que crece la tecnología de computación en la nube, aparecen unas nuevas características y se eliminan otras. Sin embargo, se pueden destacar principalmente cinco conceptos que describen este tipo de tecnología [8].

Elasticidad y habilidad de escalamiento

El Proveedor de servicios en la nube no puede saber en qué cantidad van a usar los clientes los servicios. Por ejemplo, un cliente puede usar los servicios tres veces al año en su temporada alta de negocio, mientras que otro podría usarlos diariamente debido a que tiene en la nube todas sus aplicaciones y es su entorno de desarrollo.

Por ello, el servicio debe estar disponible los 365 días del año, permitiendo escalar el uso de servicios hacia arriba, cuando en temporada alta se necesiten usar más recursos, así como hacia abajo, cuando en temporada baja exista poca demanda. La escalabilidad también responde a la posibilidad de que una aplicación pueda necesitar una mayor cantidad de recursos debido a que está siendo usada por una gran cantidad de usuarios.

La habilidad de poder escalar hacia arriba o hacia abajo los recursos es provista a través de la elasticidad. Esta última característica se podría comparar con una goma elástica. Por ejemplo, si con dicha goma se necesita sostener diez lápices, será necesario dar varias vueltas para que queden bien sujetos. En cambio, si se quieren sujetar mil lápices, la goma habrá que estirarla para poder sujetarlos todos.

Autoservicio

Cualquier usuario de la nube puede tener acceso a los recursos computacionales ofrecidos siempre cuando éste los necesite, sin ningún tipo de interacción con el personal encargado de la nube, de manera automática y unilateral.

Plataformas e interfaces

La principal plataforma con la cual se trabaja en la computación en la nube es Internet. Todos los servicios ofrecidos en la computación en la nube implican el uso tanto de clientes ligeros como de clientes pesados.

Facturación y medición
En la computación en la nube se dispone de un servicio especial interno encargado de facturar a cada cliente. Todo servicio proporcionado en la nube es medido meticulosamente, para que sea un sistema eficiente y exacto en el cobro de los servicios ofrecidos, permitiendo además optimizar el uso de cada recurso y pudiendo reasignarlos una vez que ya estén disponibles.

La facturación en la computación de la nube permite a las empresas ahorrar bastante dinero y poder emplearlo en otras fuentes de negocio. Hay que tener en cuenta que cada entidad debe tener claro las necesidades internas que tiene, para poder ahorrar lo máximo posible y aprovechar las ventajas que la computación en la nube ofrece.

Puesta en común de los recursos
Esta propiedad es una importante característica mediante la cual los Proveedores pueden lograr una gran eficiencia, proporcionando un servicio a la carta y elasticidad. Diferentes usuarios pueden compartir recursos físicos y virtuales.

2.4. Ventajas y desventajas de la computación en la nube

2.4.1. Ventajas
La utilización de los servicios ofrecidos en la computación en la nube proporciona una serie de ventajas tanto para los Consumidores como para los Proveedores, siendo algunas de ellas [9]:

- Ahorro de costes y mantenimiento. El hecho de que se pague por cada recurso que se consume ya supone un gran ahorro para una entidad. Además, no hay que preocuparse del mantenimiento de las aplicaciones, ya que el Proveedor de la nube es el encargado de ello.
- Se permite una implementación más rápida y con menos riesgos, ya que se comienza a trabajar más rápido y no es necesaria una gran inversión.
- Eliminación de distancias para el préstamo de servicios. Siempre que se disponga de un equipo con conexión y teniendo datos en la nube, se podrá acceder a los mismos sin ningún problema.

2.4.2. Desventajas
Como es lógico, este nuevo modelo de computación también tiene algunos inconvenientes, siendo algunos de ellos [9]:

- Toda la disponibilidad de las aplicaciones y de los servicios está ligada a Internet, por lo que el acceso a los mismos depende totalmente de dicha conexión.
- La información recorre varios nodos hasta que llega a su destino, por lo que si va sin cifrar, puede ser visible por terceras personas.
- Los datos sensibles del negocio no residen en las instalaciones de las empresas, lo que podría generar un contexto de alta vulnerabilidad para la sustracción de información.

2.5. Computación en la nube como tres "modelos de servicio"

El *Instituto Nacional de Estándares y Tecnología* (NIST) identifica una taxonomía simple y sin ambigüedades de tres "modelos de servicio" disponibles para los Consumidores de la nube (*Infraestructura como Servicio (IaaS)*, *Plataforma como Servicio (PaaS)* y *Software como Servicio (SaaS)*), la cual es mostrada en la *Figura 2*. Por tanto, la computación en la nube basa su arquitectura haciendo una separación entre hardware, plataforma y aplicaciones, quedando las siguientes capas [6]:

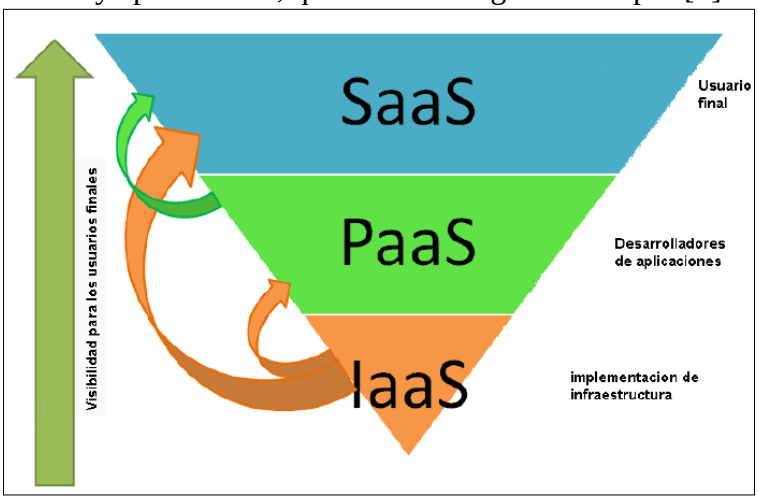


Figura 2. Modelos de servicio de la computación en la nube

Software como servicio (*SaaS*)

Se encuentra en la capa más alta y consiste en la entrega de aplicaciones completas como un servicio. El Proveedor se encarga de dar servicio a multitud de clientes a través de la red sin que éstos tengan que instalar ningún software adicional. La distribución de la aplicación tiene el modelo de uno a muchos, es decir, se elabora un producto que es usado por varios clientes. Los Proveedores son los responsables de la disponibilidad y funcionalidad de sus servicios, no dejando de lado las necesidades de los clientes. Las actividades son gestionadas desde algún lugar central, en vez de hacerlo desde la sede de cada cliente, permitiendo a los mismos el acceso remoto a las aplicaciones vía web. Las actualizaciones son centralizadas, eliminando la necesidad de descargar parches por parte de los usuarios finales.

Plataforma como servicio (*PaaS*)

La siguiente capa en orden descendente es *PaaS* (*Plaform as a Service*), cuyo objetivo se centra en un modelo que proporciona un servicio de plataforma con todo lo necesario para dar soporte al ciclo de planteamiento, desarrollo y puesta en marcha de aplicaciones y servicios web. El Proveedor es el encargado de escalar los recursos en caso de que la aplicación lo requiera, de que la plataforma tenga un rendimiento óptimo, de la seguridad de acceso, etc. Con *PaaS* el cliente únicamente se enfoca en desarrollar, depurar y probar ya que la herramienta necesaria para el desarrollo de software es ofrecida a través de Internet.

Infraestructura como servicio (*IaaS*)
IaaS (*Infrastructure as a Service*) corresponde a la capa más baja. La idea principal es la de hacer uso externo de servidores para espacio en disco, base de datos así como de tiempo cómputo para de esta manera evitar tener un servidor local y toda la infraestructura necesaria para la conectividad y mantenimiento dentro de una organización. Con este tipo de modelo se tiene una solución en la que se paga solamente por consumo de recursos usados.

2.6. Principales servicios de almacenamiento en la nube

En la actualidad, hay muchas empresas que están apostando por la prestación de servicios de almacenamiento en la nube y adaptando su *software* para ser usado de esta manera. A continuación se van a citar las más conocidas, ofreciendo una breve descripción de cada una de ellas [5].

***Dropbox*[1]**
Se trata de una empresa que ofrece servicios de almacenamiento remoto y de copias de seguridad. Tiene una versión gratuita de 2GB de capacidad, pudiendo guardar cualquier tipo documento accediendo desde cualquier dispositivo.

***Google Drive*[2]**
Es un servicio de alojamiento de archivos ofrecido por *Google*. Cada usuario cuenta con 15GB de espacio gratuito para almacenar sus archivos, ampliables mediante pago. Es accesible mediante su página web desde ordenadores y dispone de aplicaciones para *iOS* y *Android*.

***OneDrive*[3]**
Este servicio es prestado por la empresa *Microsoft*, consistiendo en el almacenamiento remoto de archivos en línea para poder verlos con un navegador Web. En un principio aporta 7GB libres, pudiendo subir archivos de hasta 2GB.

***iCloud*[4]**
Este servicio es prestado por la empresa *Apple* integrado en todos sus dispositivos, permitiendo realizar copias de seguridad de todos los datos contenidos en ellos. Se tiene un máximo de 5GB, pudiéndose ampliar previo pago.

[1] https://www.dropbox.com
[2] https://drive.google.com
[3] https://onedrive.live.com
[4] https://www.icloud.com

Capítulo 3
Estudio de la seguridad en los entornos *cloud*

3.1. Introducción

La idea de computación en la nube es hoy en día bien conocida y ampliamente utilizada. Según *NIST* [7], [10], como ya se ha comentado en el apartado 2.3, hay cinco características esenciales de la computación en la nube: autoservicio bajo demanda, acceso completo a la red, puesta en común de los recursos, elasticidad y un servicio medido. Cada una de estas características proporciona recursos computacionales más eficientes y flexibles en beneficio de una amplia gama de Consumidores. Al mismo tiempo, cada característica implica un desafío que compromete la seguridad.

Individuos y pequeñas empresas pueden eliminar la gran carga que supone llevar a cabo buenas prácticas de seguridad, pudiendo confiar en el Proveedor de la nube para que se encargue de estas tareas. Este escenario tiene sentido si el Consumidor confía en el Proveedor para que éste sea el responsable de sus datos. Sin embargo, algunos usuarios de la nube no tienen plena confianza en su Proveedor. En esta situación, los usuarios necesitan implementar y administrar la seguridad de sus propios datos de tal manera que todavía puedan beneficiarse de los servicios que la nube ofrece. Esta tensión de confiar la seguridad de los datos a un tercero, y a la vez poder mantener el control de los mismos, es el gran desafío de la seguridad en la computación en la nube.

La criptografía es ahora una tecnología dominante usada para mantener la confidencialidad y la integridad de los datos que son transmitidos o almacenados en forma electrónica. Algoritmos estandarizados, autenticación de mensajes y firmas digitales se aplican de forma rutinaria para asegurar que los datos sólo pueden ser leídos por los destinatarios autorizados, para verificar que los datos no han sido alterados y para asegurar que los remitentes son los responsables de las firmas realizadas. Estos algoritmos se pueden aplicar a los datos que van a ser enviados a la nube. El problema fundamental es qué parte controla las claves necesarias para descifrar los datos y para realizar las comprobaciones de integridad. Si las claves son conocidas por el Proveedor, entonces el Consumidor tendrá que confiar en que no viole la confidencialidad de los datos, por ejemplo. Por otro lado, si el Consumidor guarda las claves, esto devuelve la carga de gestión a dicho Consumidor, pero podrá tener un control total sobre sus datos.

3.2. Requisitos de seguridad para un sistema de computación en la nube

Se pueden diferenciar tres problemas fundamentales a la hora de asegurar la seguridad de la información: confidencialidad, integridad y disponibilidad. Estos servicios de

seguridad tienen por objeto garantizar que los datos están disponibles sólo para las partes correctas, de que la información no es alterada y de que se puede acceder cuando sea necesario a dicha información por cualquiera de las dos partes autorizadas. En la computación en la nube surgen nuevas amenazas de seguridad debido a la nueva arquitectura de procesamiento de datos. Se van a explorar estas amenazas partiendo de las características principales de la computación en la nube mencionadas anteriormente [10].

Autoservicio bajo demanda. Esta propiedad permite a los Consumidores utilizar el almacenamiento y el procesamiento de datos proporcionados por el Proveedor siempre que sea necesario. Esto significa que los datos de usuario van a ser almacenados y procesados en los servidores del Proveedor. Si dichos datos tienen valor comercial (por ejemplo, un contrato), entonces el Consumidor deberá realizar una evaluación de riesgos por la posibilidad de que los datos se vean comprometidos o alterados. Si el Consumidor no tiene la suficiente confianza en el Proveedor, se tendrá que encargar de proteger sus datos criptográficamente para evitar que sean leídos o modificados por el Proveedor.

Acceso completo a la red. Esta propiedad permite a los Consumidores acceder a los servicios de la nube desde una gran variedad de plataformas. El acceso remoto requiere una autenticación del usuario al Proveedor, y también del Proveedor al usuario. El uso de Internet o de redes inalámbricas para dicho acceso abre la posibilidad de un espionaje de las comunicaciones entre el Consumidor y el Proveedor. Esto puede requerir que la información intercambiada para realizar esta autenticación tenga que ser cifrada.

Puesta en común de los recursos. Esta propiedad describe la puesta en común de los recursos del Proveedor a los diferentes Consumidores. Esta es una importante manera por la que los Proveedores pueden lograr una gran eficiencia, proporcionando un servicio a la carta y elasticidad. Diferentes usuarios pueden compartir recursos físicos y virtuales. Con diversos mecanismos se puede asegurar que los Consumidores sólo pueden acceder a los recursos a los que tienen acceso. Esta propiedad de nuevo sugiere la necesidad del cifrado de los datos de usuario.

Elasticidad. Esta propiedad permite a los Consumidores ampliar o reducir rápidamente el uso de los recursos de la nube. Debido a que la escala de servicios en la nube puede proporcionar la apariencia de recursos infinitos para el Consumidor, es una razón atractiva para mudarse a la computación en la nube. Por otra parte, los Consumidores pueden reducir rápidamente sus requerimientos cuando no necesitan usar tantos recursos de la nube. Esta flexibilidad se puede lograr mediante el intercambio de recursos, que conducen a los problemas de seguridad descritos en el punto anterior.

Servicio a medida. Esta propiedad permite a los Consumidores pagar sólo por los recursos que utiliza.

En la *Tabla 1* se resumen las amenazas de seguridad que se han identificado anteriormente. Se puede observar que el cifrado de los datos de usuario es una forma

útil para lograr confidencialidad en dichos datos y esto se puede complementar con comprobaciones de integridad o firmas digitales para evitar sufrir modificaciones.

Amenaza	Agente de amenaza	Medidas de mitigación
Espionaje de información	Proveedor Otros usuarios de la nube Hackers	Cifrado
Modificación de los datos	Proveedor Otros usuarios de la nube Hackers	Comprobaciones de integridad Firmas digitales
Disponibilidad	Proveedor	Redundancia Pruebas de almacenamiento
Movimiento de datos	Proveedor	Pruebas de localización

Tabla 1. Amenazas de seguridad en los Consumidores en la computación en la nube

3.3. Claves criptográficas en entornos *cloud*

Como ya se ha podido ver, la criptografía juega un papel fundamental a la hora de poder garantizar seguridad en la computación en la nube. Por tanto, se usan diferentes tipos de claves que son necesarias para poder utilizar los distintos algoritmos criptográficos. Estos tipos de claves se van a detallar en el siguiente apartado.

3.3.1. Tipos de claves

Las claves criptográficas se dividen en dos grandes categorías:

1. **Clave secreta**. Clave que se utiliza generalmente para 1) realizar el cifrado/descifrado utilizando algoritmos criptográficos simétricos, y/o 2) para proporcionar integridad de datos mediante códigos de autenticación de mensajes. Una clave secreta también se llama clave simétrica, ya que se requiere la misma clave para el cifrado y descifrado.
2. **Par de claves Pública/Privada.** Un par de claves relacionadas matemáticamente utilizadas en la criptografía asimétrica para la autenticación, firma digital, o el establecimiento de claves. Como su nombre indica, la clave privada es utilizada por el propietario del par de claves, se mantiene en secreto, y debe ser protegida en todo momento, mientras que la clave pública puede ser publicada y utilizada por la parte que confía en completar el protocolo o invertir las operaciones realizadas con la clave privada [11]. Este tipo de claves se utiliza en el cifrado asimétrico.

A partir de estas categorías generales, se puede determinar los tipos de clave más utilizados en un entorno de *Cloud Computing*. Esto no quiere decir que una implementación en la nube no puede tener otros tipos de claves. Los posibles usos que se pueden dar a estas claves en estos tipos de entornos son los siguientes:

1. **Autenticación con par de claves Pública/Privada.** Este par de claves es utilizado por una de las partes (peer, cliente o servidor) para autenticar a la otra parte.

2. **Firma con par de claves Pública/Privada.** La clave privada del par de claves es usada por una parte para la firma digital del mensaje/datos, y la correspondiente clave pública es usada para poder verificar la firma.
3. **Establecimiento de clave con par de claves Pública/Privada.** Este par de claves se utiliza para establecer con seguridad una clave entre las dos partes.
4. **Cifrado/descifrado con clave simétrica**. Una clave simétrica es usada para cifrar y descifrar mensajes/datos.

3.3.2. Estados de la clave

Una clave simétrica o un par de claves pública/privada pueden someterse a los siguientes estados [11].

- **Generación (*Generation*).** Una clave simétrica o un par de claves pública/privada son generadas cuando son necesarias.
- **Activación (*Activation*)**. Una clave simétrica o un par de claves pública/privada son activadas cuando son necesitadas para su uso.
- **Desactivación (*Deactivation*)**. Una clave simétrica o un par de claves pública/privada son desactivadas cuando ya no son necesarias para la aplicación de la protección a los datos.
- **Suspensión (*Suspension*).** Una clave puede ser suspendida de su uso por varias razones, debido a que no se conoce el estado de la clave o debido a que el propietario de la clave no se encuentre disponible temporalmente.
- **Expiración (*Expiration*).** Una clave puede ser expirada debido a que su periodo de uso ha finalizado.
- **Destrucción (*Destruction*).** Una clave es destruida cuando ya no es necesaria.
- **Archivada (*Archival*).** Una clave puede ser archivada cuando no es necesitada para su uso normal, pero puede ser necesaria posteriormente.
- **Revocación (*Revocation*).** La revocación aplica principalmente a las claves públicas, aunque también puede aplicar a la correspondiente clave privada.

A continuación, en la *Figura 3* se expone el diagrama de estados para los estados de una clave.

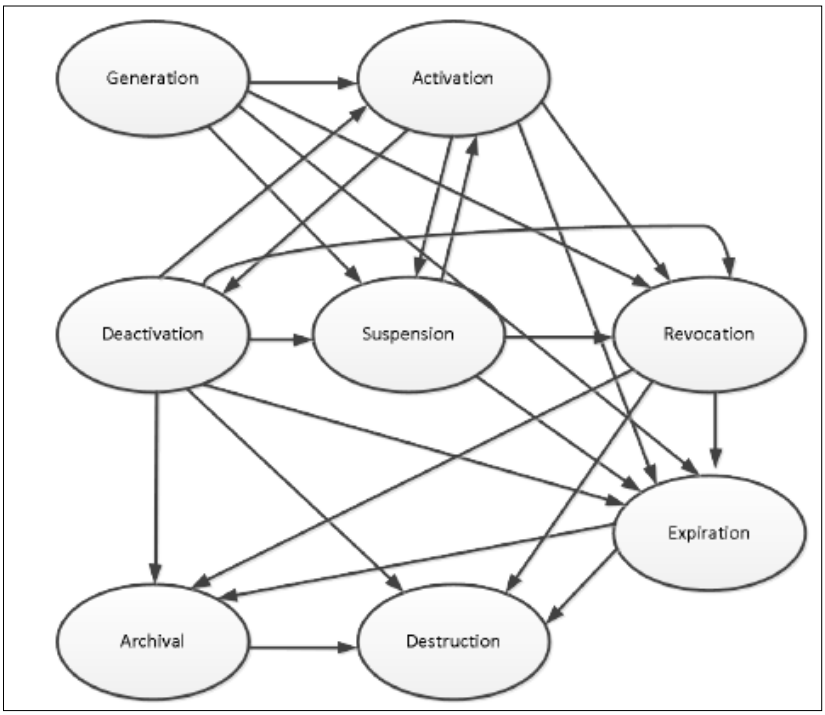


Figura 3. Diagrama de estados para los estados de una clave

3.3.3. Funciones de administración de claves

Las siguientes funciones de administración de claves son las más importantes [11]:

- **Generar clave.** La generación de una clave de calidad es un aspecto crítico para la seguridad.
- **Generar parámetros de dominio**. Algunos algoritmos requieren la generación de parámetros de dominio antes de la generación de claves.
- **Enlazar clave y metadatos.** Una clave puede tener datos asociados, como puede ser el período de tiempo de uso, limitaciones de uso, parámetros de dominio y servicios de seguridad para lo cual se utiliza, como son la autenticación del origen, la integridad y la protección de la confidencialidad. Esta función proporciona la seguridad de que la clave está asociada con los metadatos correctos.
- **Enlazar una clave a una persona.** El identificador del individuo o entidad que posee una clave se considera como parte de los metadatos de la clave, y esta asociación es lo suficientemente crítica como para ser tratada por una función independiente.
- **Activar clave.** Esta función realiza la transición de una clave al estado activado. Se suele realizar al crear la clave.
- **Desactivar clave.** Esta función se realiza generalmente cuando una clave no es necesaria para la aplicación de la protección criptográfica. Por ejemplo, cuando una clave ha caducado o se sustituye por otra clave.

- ***Backup* de la clave.** Una clave puede ser respaldada por su propietario, por la infraestructura de gestión de claves o por un tercero con el fin de reconstruir la clave cuando por ejemplo, se ha destruido accidentalmente.
- **Recuperación de la clave**. Esta función es complementaria a la función anterior y se invoca cuando la clave no está disponible por alguna razón y es requerida por las partes autorizadas.
- **Modificación de metadatos.** Esta función es llamada cuando se necesitan modificar los metadatos asociados a una clave. La renovación de un certificado de clave pública es un ejemplo de esta función, donde se cambia el período de validez para la clave pública.
- ***Rekey*.** Esta función se utiliza para reemplazar una clave existente por una clave nueva.
- **Suspensión de clave.** Esta función se utiliza cuando se deja de utilizar temporalmente una clave, y puede ser invocada si el estado de una clave es indeterminado o si el propietario de la clave desea suspender temporalmente su uso.
- **Restauración de clave**. Esta función es utilizada para restaurar una clave suspendida después de comprobar que su estado es seguro.
- **Revocación de clave.** Esta función es utilizada con el fin de informar a las partes de confianza para dejar de usar una clave pública.
- **Almacenamiento de clave.** Esta función es utilizada para archivar una clave después de que haya sido desactivada, expirada o comprometida.
- **Destrucción de clave.** Esta función se utiliza para destruir una clave cuando ya no se va a volver a usar.

3.4. Garantías para el almacenamiento remoto de datos

Los Consumidores que desean utilizar entornos *cloud*, típicamente desean colocar una gran cantidad de sus datos en los servidores de la nube. Parte de estos datos puede que no se vayan a utilizar durante períodos prolongados, pero su disponibilidad cuando sean necesarios puede ser un aspecto crítico. Quiera o no el Consumidor procesar esos datos, es posible que se desee comprobar de vez en cuando que los datos siguen estando disponibles. Una manera de hacer esto es recuperar una copia de los datos en la nube y revisarlos usando una suma de comprobación basada en una función hash criptográfica. Sin embargo, esta forma de realizar dicha comprobación es muy ineficiente cuando los datos son muy grandes. La comunidad criptográfica propone un mecanismo *online* de la verificación de la disponibilidad de los datos examinando una porción pequeña de los mismos combinando un procesamiento criptográfico.

3.4.1. Pruebas de almacenamiento y pruebas de *recuperabilidad*

La aplicación de pruebas de *recuperabilidad* en computación en la nube parece haber sido reconocido por primera vez por *Juels* y *Kaliski* [12]. Su idea era primero cifrar la información y luego insertar bloques de datos llamados "*sentinels*" en la información cifrada en puntos aleatorios (los *sentinels* son generados usando una función criptográfica cuyos valores no se pueden distinguir de los datos cifrados sin conocer las

entradas de la función). De esta forma se prepara un archivo que se envía a continuación al servidor de nube para el almacenamiento. El Proveedor sólo ve un fichero de bits aleatorios, y no puede distinguir entre qué partes de los datos corresponden con la encriptación original y qué partes han sido introducidas aleatoriamente (*sentinels*). El dueño de los datos puede verificar si los datos están intactos, pidiendo una selección aleatoria de los *sentinels* que son devueltos especificando sus posiciones que sólo el dueño sabe, y puede verificar si los *sentinels* son devueltos correctamente por el servidor. Como el Proveedor no sabe dónde se encuentran los *sentinels*, cualquier cambio en los datos puede ser detectado.

3.4.2. Pruebas de localización

La localización de los datos del Consumidor en la nube puede ser un problema de seguridad importante. Cuando un Proveedor garantiza que los datos se almacenan dentro de una cierta área geográfica puede que los Consumidores puedan obtener cierta seguridad acerca de ello. La idea básica para lograr esa seguridad es medir cuánto tiempo se necesita para que los datos sean devueltos al usuario desde una ubicación concreta. Cuanto más lejos se almacenen los datos, más tiempo se tardarán en transmitir. Estas mediciones deben llevarse a cabo con un alto grado de precisión ya que la información se mueve muy rápido en las comunicaciones electrónicas.

Un ejemplo de prueba de localización es el uso de protocolos delimitadores de distancia. Estos protocolos siempre implican una fase de temporización en la que el verificador emite un "desafío" y el Proveedor responde. Se requiere que el Proveedor responda dentro de un plazo límite que va a depender de la distancia entre ambos. El aspecto de cifrado es usado para asegurar que las respuestas vinieron del Proveedor requerido y no de otra entidad que se hace pasar por ese Proveedor.

En la nube, se puede usar este método para verificar la exactitud del continente, o incluso el país en el que se encuentra los datos. También hay una serie de incertidumbres que pueden complicar dicha medición. Por ejemplo, al verificar los datos almacenados se pueden sufrir algunos retrasos debidos a que el servidor tarda un tiempo en acceder al disco y al sector en el cual se encuentran los bloques de archivos necesarios. Además, también puede haber un retraso en la comunicación vía Internet.

3.4.3. Pruebas de redundancia

Otra propiedad que puede ser de interés para los Consumidores es comprobar que se han realizado copias de seguridad adecuadamente. Una vez más, al principio puede parecer que tal propiedad es imposible de verificar. Sin embargo, las técnicas por temporización pueden proporcionar una forma de lograr esta verificación. *Bowers* [13] diseñó un protocolo para demostrar que los datos son almacenados en más de un disco, es decir, que se hacen copias de seguridad. Esto se puede hacer observando el tiempo que se tarda en obtener partes aleatorias de datos, y es debido a las limitaciones físicas sobre el almacenamiento en disco, ya que se puede acceder a los datos de una forma más rápida si hay dos copias en discos diferentes, permitiendo el acceso simultáneo a cada uno de

ellos. Cabe destacar que este protocolo sólo es aplicable para las copias que se encuentran almacenadas en el mismo centro de datos.

Capítulo 4
Descripción del proyecto software implementado

4.1. Introducción

En el capítulo anterior, se han mencionado y estudiado varios aspectos que están relacionados con la seguridad en entornos *cloud*. Sin embargo, no todos estos elementos se van implementar en el proyecto realizado, ya que hay que tener presente que el objetivo principal del mismo es la distribución de claves criptográficas simétricas en este tipo de entorno. Además de esta última tarea, también se ha tenido que utilizar y gestionar tanto claves simétricas como asimétricas, descritas en el apartado *3.3.1*. También ha sido importante en el proyecto la comprobación de la integridad de los activos subidos, con el fin de poder verificar que no son manipulados por un tercero no autorizado.

Tal y como se ha descrito en la introducción del presente trabajo, el objetivo de este proyecto es ilustrar el problema de distribución de claves criptográficas simétricas, mediante la implementación de un prototipo *OpenSource*, desarrollado para dispositivos móviles con sistema operativo *Android* a partir de la versión de *firmware* 2.3.6, capaz de subir, descargar y compartir cualquier tipo de archivo cifrado entre diferentes cuentas de *Dropbox*. Los usuarios deben tener presente que los ficheros subidos a través de la aplicación desarrollada no pueden ser accedidos a través de cualquier otro acceso a *Dropbox* que no sea el de la misma aplicación. Es importante tener en cuenta que se ha partido del Trabajo de Fin De Grado "*Gestión de documentos en entornos colaborativos mediante un cliente Android*", realizado por Ignacio del Pozo Martínez.

Para cumplir con este objetivo, primero se va a detallar la infraestructura cliente-servidor elegida. A continuación, se presentará la planificación seguida durante la realización del proyecto. Luego se va a proporcionar un listado de requisitos funcionales y no funcionales que el prototipo debe cumplir además de describir todos los aspectos relevantes acerca del diseño, la funcionalidad y la implementación del mismo. Por último, se terminó documentando el plan de pruebas llevado a cabo.

4.2. Infraestructura cliente-servidor elegida

4.2.1. *Dropbox* como servidor

Dropbox es un servicio de alojamiento de archivos en la nube, que permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores, además de compartir archivos y carpetas con otros [14].

Es una aplicación multiplataforma, compatible con *Windows*, *Linux* y *Mac*, y con dispositivos móviles *Android*, *Windows Phone*, *Blackberry* e *IOS*.

Hay tres tipos de cuenta, la cuenta gratuita "*Free*" que es la primera, la segunda "*Pro*" y la tercera empresarial "*Business*" que son de pago. Las diferencias entre estas tres cuentas están en la cantidad de espacio que se puede utilizar, ya que la gratuita tiene una capacidad inicial de 2GB pudiendo llegar hasta 18GB, mientras que en las cuentas "*Pro*" y "*Business*" se pueden alcanzar los 500GB y 1TB respectivamente.

Dropbox se anuncia como un servicio totalmente seguro, diciendo que ni siquiera sus empleados tienen acceso a los datos guardados. No obstante, obligan al usuario a confiar sus datos en un servidor que no conocen, de ahí la necesidad de subir los activos cifrados.

Como se ha expuesto en el apartado *2.6*, existen otros sistemas de disco duro virtuales ampliamente conocidos. Se ha optado por *Dropbox* debido a su facilidad de uso, a la gran cantidad de usuarios activos que tiene actualmente y a su fácil integración en plataformas como *Android*. No obstante, la funcionalidad implementada en la aplicación desarrollada se podría adaptar fácilmente a cualquiera de las otras alternativas.

Es interesante conocer las ventajas e inconvenientes existentes al usar un servicio como *Dropbox*.

Ventajas

- Fácil de usar y de configurar gracias a su interfaz sencilla.
- Archivos siempre sincronizados entre todos los dispositivos vinculados a una determinada cuenta de *Dropbox*.
- Se ahorran recursos de hardware, ya que se usan recursos de los servidores.

Inconvenientes

- El espacio de almacenamiento es limitado, por lo que hay que pagar una tarifa si se necesitan más GB.
- Los archivos se encuentran en un espacio virtual que no puede ser controlado por el usuario.
- Varias personas no pueden modificar documentos simultáneamente en tiempo real, lo que dificulta el trabajo colaborativo a través de este servicio.

4.2.2. Dispositivo móvil *Android* como cliente

En la actualidad, el uso de un dispositivo móvil es algo cotidiano en gran parte de la sociedad. Esto es debido a que, sobre todo con la llegada del *smartphone*, se puede tener un dispositivo que es prácticamente como un ordenador. Su pequeño tamaño y las prestaciones que ofrece, hacen que sea uno de los aparatos electrónicos más usados del mercado.

Cada dispositivo móvil tiene un sistema operativo, siendo los más destacados los siguientes:

- *Bada*.
- *Windows Mobile*.
- *Android*.
- *BlackBerry*.
- *iOS*.
- *Symbian*.

En la *Figura 4* se puede observar la cuota de mercado de cada sistema operativo, desde el año 2007 al año 2012.

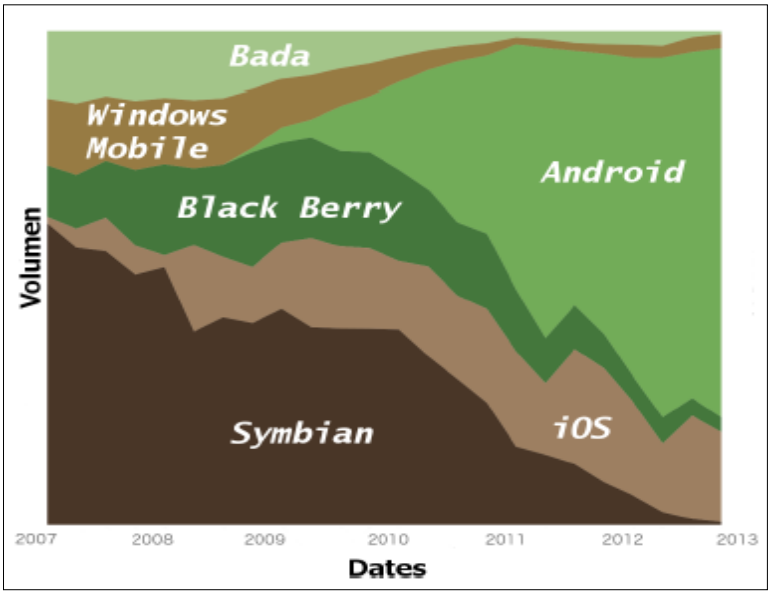


Figura 4. Principal sistema operativo de *smartphones* según la cuota de mercado desde 2007 a 2012 [15]

Debido a que el sistema operativo *Android* abarca aproximadamente el 80% del mercado, se ha optado por implementar la aplicación en esta plataforma, que dispone de un entorno de desarrollo para poder programar sus aplicaciones. No obstante, es cierto que lo ideal hubiera sido implementar la aplicación en los distintos sistemas operativos mencionados anteriormente, pero debido a la restricción de tiempo, se ha optado por la elección de la plataforma *Android*.

Android es un sistema operativo además de una plataforma de *software* basada en el núcleo de *Linux*. Diseñada en un principio para dispositivos móviles, *Android* permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptados por *Google* mediante el lenguaje de programación *Java*.

Android es una plataforma de código abierto. Esto quiere decir que cualquier desarrollador puede crear y desarrollar aplicaciones [16].

Android ha tenido esa cuota creciente de mercado debido a que es un sistema operativo que está presente en multitud de dispositivos a parte de los móviles, teniendo precios

asequibles los que son de gama baja, y precios más elevados los dispositivos de gama alta.

4.3. Planificación

En este apartado se va a detallar a través de un diagrama de *Gantt*, la planificación seguida a lo largo de la realización del proyecto. Cabe destacar que la planificación mostrada es la actualizada en la parte final del proyecto. El diagrama sigue un tipo de calendario laborable, dedicando una media de tres horas diarias. Este diagrama se puede observar en el *Anexo A. Diagrama de Gantt*, siendo explicado a continuación.

La planificación seguida a lo largo del proyecto se puede diferenciar en cinco fases:

Documentación 06/02/14 – 20/03/14
Durante esta fase, se dedicó todo el tiempo a leer documentación acerca de la computación en la nube, y especialmente sobre el estado actual de la seguridad en estos entornos. Se leyeron diversas publicaciones, que han servido como punto de partida y que se han nombrado a lo largo del presente documento. Además, el día 20 de Marzo de 2014, se asistió al *Hotel NH Eurobuilding* de Madrid a una conferencia titulada "*Ciberseguridad basada en la nube*" [17], en la que empresas como *Check Point*, *Cisco*, *FireEye*, *Palo Alto*, *Trend Micro* y *Zscaler* expusieron sus propuestas de seguridad en la nube. Como usuarios, fueron representantes del *Grupo Banco Sabadell* y de *Mediaset*, contando a todos los presentes la gran preocupación que tenían acerca de llevar sus datos más críticos a la nube.

Análisis y Diseño 21/03/14 – 01/04/14
Durante esta fase, que duró ocho días, se llevo a cabo la toma de requisitos de la aplicación así como su diseño.

Programación 02/04/14 – 21/05/14
Después de haberse documentado lo suficiente como para entender el problema que había que resolver, se pasó a la fase de programación.

Primero, se estuvo durante quince días (02/04/14 – 22/04/14) conociendo toda la parte criptográfica de *Android*, desconocida hasta ese momento. Se hicieron simples tutoriales de cifrados y descifrados de cadenas, luego de textos más grandes y finalmente de ficheros. Por último, se llevó a cabo la implementación de un formato de paquete que permitiera el intercambio de una clave simétrica entre varios usuarios.

A continuación, se realizó la integración de la API de *Dropbox* en *Android*, además de hacer pruebas con la misma subiendo y bajando ficheros a través de una cuenta de este tipo de servicio. Esta tarea tuvo una duración de aproximadamente cuatro días (23/04/14 – 28/04/14).

La siguiente tarea fue la implementación de un explorador de ficheros de *Dropbox*, que es utilizado cuando un usuario quiere subir o bajar un fichero, o también unirse o abandonar una carpeta compartida. El explorador se terminó en un plazo de tres días (29/04/14 – 01/05/14).

El siguiente paso fue la codificación de la funcionalidad necesaria para permitir a un usuario unirse o abandonar una carpeta compartida, que será explicada más detalladamente en los apartados *4.7.6.3* y *4.7.6.4*. Se pudo terminar pasados cuatro días (02/05/14 – 07/05/14).

Una vez que se llegó a este punto, se realizó la codificación más complicada, correspondiente a la subida y bajada de ficheros cifrados, con la opción de poder compartirlos. En este momento se adaptó el código elaborado al principio de la fase de Programación, que era el encargado de poder distribuir una clave simétrica entre varios usuarios. A esta parte de la fase actual se le dedicaron siete días (08/05/14 – 16/05/14).

Por último, se implementaron una serie de servicios que se describirán en apartados posteriores, encargados de asegurar que los ficheros depositados en *Dropbox* han sido correctamente firmados por el usuario que los subió y de actualizar los ficheros de las carpetas compartidas, cuando se abandonan o se unen nuevos usuarios, además de comprobar la posible eliminación por parte de un tercero de los ficheros subidos. Estos servicios fueron implementados en tres días (19/05/14 – 21/05/14).

Pruebas (22/05/14 – 28/05/14)
Una vez que se acabó el desarrollo de la aplicación, se pasó a la fase de pruebas, con el fin de comprobar que funcionaban correctamente todas las características que se habían establecido en un principio y que la aplicación debía cumplir. Para ello, se diseñó un plan de pruebas, que se encuentra más detallado en la sección *4.8*. Se realizó cada prueba, documentándola y analizando los resultados obtenidos.

Memoria (29/05/14 – 30/06/14)
Cuando se terminó de implementar y de comprobar la funcionalidad de la aplicación, se empezó a redactar la memoria. Esta redacción se ha realizado durante catorce días (29/05/14 – 17/06/14), y una vez terminada, se procedió a su corrección durante nueve días (18/06/14 – 30/06/14).

4.4. Catálogo de requisitos

En este apartado se van a detallar el conjunto de especificaciones funcionales y no funcionales que debe cumplir la aplicación desarrollada.

4.4.1. Requisitos Funcionales

Los requisitos funcionales son aquellos que tienen que ver con el comportamiento de la aplicación, que están directamente relacionados con el funcionamiento de la misma.

RF1. Vincularse con cualquier cuenta *Dropbox* previamente creada.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Vincular con *Dropbox*".
2. Se abrirá el navegador, en el cual el usuario introducirá las credenciales de su cuenta de *Dropbox* y pulsará sobre "Iniciar sesión".

3. Se le pedirá en la siguiente pantalla que su cuenta de *Dropbox* dé permisos a la aplicación "*EncryptedCloud* API". El usuario deberá pulsar sobre "Permitir", realizándose el vínculo con su cuenta de *Dropbox*.

RF2. Desvincularse de cualquier cuenta *Dropbox* previamente creada y vinculada.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Desvincular con *Dropbox*".

RF3. Subir cualquier fichero del dispositivo móvil a una cuenta *Dropbox*.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Subir Fichero".
2. A continuación, el usuario seleccionará el fichero de su dispositivo móvil que quiere subir a su cuenta de *Dropbox*.
3. Por último, seleccionará la carpeta de su cuenta de *Dropbox* en la cual quiere subir su fichero previamente cifrado.

RF4. Descargar cualquier fichero contenido en una cuenta *Dropbox*.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Descargar Fichero".
2. A continuación, el usuario seleccionará el fichero de su cuenta *Dropbox* que quiere descargarse.
3. Al seleccionar dicho fichero se le guardará descifrado en su dispositivo, en una carpeta llamada *EncryptedCloud*.

RF5. Unirse a una carpeta compartida previamente creada en una cuenta *Dropbox*.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Unirse Carpeta Compartida".
2. Se le mostrarán al usuario todas las carpetas compartidas creadas en su cuenta de *Dropbox*.
3. El usuario seleccionará la carpeta a la cual se quiere unir.

RF6. Abandonar una carpeta compartida previamente creada en una cuenta *Dropbox*.

1. Desde la pantalla principal de la aplicación, el usuario pulsará sobre el botón "Abandonar Carpeta Compartida".
2. Se le mostrarán al usuario todas las carpetas compartidas creadas en su cuenta de *Dropbox*.
3. El usuario seleccionará la carpeta que quiere abandonar.

RF7. Registrar a un usuario en la aplicación.

1. Desde la pantalla inicial de la aplicación, el usuario pulsará sobre el botón "*Login*".
2. A continuación, pulsará sobre el botón "Nuevo Usuario".
3. Seguidamente, deberá rellenar el campo usuario y repetir la contraseña dos veces, por cuestiones de seguridad.
4. Por último, pulsará sobre el botón "Aceptar" para crear la nueva cuenta en la aplicación.

RF8. *Loguear* a un usuario previamente registrado en la aplicación.

1. Desde la pantalla inicial de la aplicación, el usuario pulsará sobre el botón "*Login*".
2. A continuación, deberá rellenar el campo usuario y contraseña.
3. Por último, pulsará sobre el botón "Aceptar" para acceder con su cuenta a la aplicación.

RF9. No permitir la creación de dos usuarios con el mismo nombre.

RF10. Monitorizar la posible eliminación o modificación de ficheros subidos a una cuenta de *Dropbox*.

4.4.2. Requisitos No Funcionales

Los requisitos no funcionales son aquellos que definen características de la aplicación que no son necesarias para su funcionamiento.

RNF1. Se deberá permitir modificar la contraseña de una cuenta de usuario previamente creada.

1. Desde la pantalla inicial de la aplicación, el usuario pulsará sobre el botón "*Login*".
2. A continuación, pulsará sobre el botón "Modificar Contraseña".
3. Seguidamente, deberá rellenar el campo usuario, la contraseña antigua y repetir la contraseña nueva dos veces, por cuestiones de seguridad.
4. Por último, pulsará sobre el botón "Aceptar" para modificar la contraseña de dicha cuenta.

RNF2. La aplicación debe ser fácil de usar, de forma que cualquier persona pueda tener acceso a ella. La deberán probar distintas personas con diferentes conocimientos informáticos para verificar este requisito.

RNF3. La aplicación debe realizar un *backup* de la base de datos local después de autenticarse un usuario.

RNF4. La aplicación será capaz de importar una base de datos local.

1. Siempre que anteriormente se haya hecho un *backup* de la base de datos local, el usuario podrá importarlo a través del botón "Importar Base De Datos" de la pantalla inicial.

RNF5. La aplicación estará disponible en varios idiomas, según el establecido en el dispositivo móvil. Los idiomas disponibles son castellano, búlgaro, catalán, checo, danés, alemán, griego, inglés, estonio, finlandés, francés, húngaro, italiano, lituano, letón, noruego, polaco, portugués, y rumano.

RNF6. La aplicación deberá funcionar en cualquier dispositivo *Android* a partir de la versión de *firmware* 2.3.6.

RNF7. La aplicación deberá ser modular, para tener la posibilidad de poder reutilizar parte de ella en cualquier otro proyecto. Para cumplir con esta modularidad, el código será dividido en distintos paquetes, cuyos nombres siempre empezarán por *es.uam.eps.tfg.encryptedcloud*.

RNF8. La aplicación deberá estar documentada detalladamente, con el fin de poder facilitar su mantenimiento e integración con otros sistemas.

4.5. Diseño

En esta sección se van a presentar dos diagramas que fueron imprescindibles en el momento de realizar el diseño de la aplicación software implementada.

4.5.1. Diagrama entidad-relación de la base de datos

Un diagrama entidad-relación es una herramienta para el modelado de datos que permite representar las distintas entidades de una base de datos así como sus interrelaciones y sus propiedades [18].

El diagrama entidad-relación diseñado para la aplicación es el mostrado en la *Figura 5*.

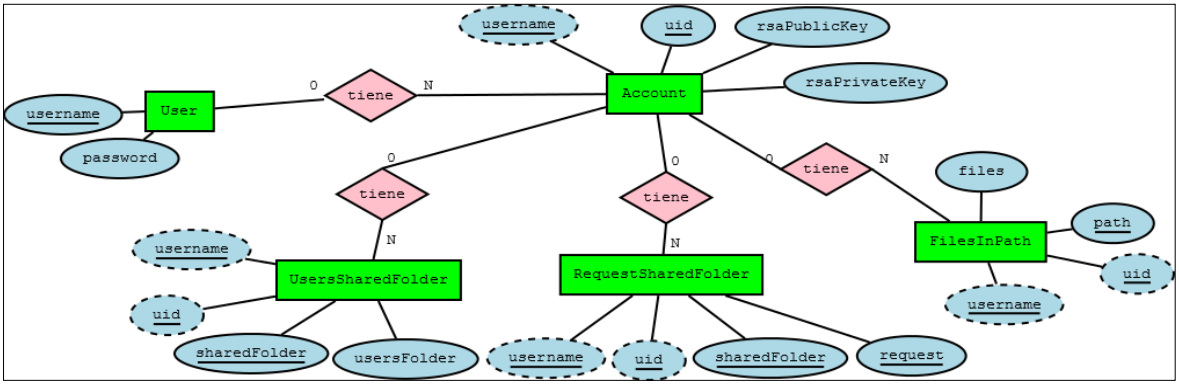


Figura 5. Diagrama entidad-relación

En la base de datos diseñada, se distinguen cinco entidades distintas.

La primera entidad, ***User***, es la encargada de almacenar a todos los usuarios registrados en la aplicación. Para ello, tendrá dos atributos. El primero de ellos es *username*, que almacena el nombre de usuario y es la clave primaria de la entidad, por lo que no pueden existir dos usuarios con el mismo nombre. El otro atributo es *password*, en el que se almacenará el *hash* de la contraseña con la que se ha registrado el usuario. En la *Figura 6* se puede observar un registro de la entidad ***User***.

username	password
alex	ca978112ca1bbdcafac231b39a23dc4da786eff8147c4e72b9807785afee48bb

Figura 6. Registro de la entidad *User*

La segunda entidad es ***Account***, en la que se almacenarán las cuentas con las que se ha vinculado cada usuario. La clave primaria de esta entidad está formada por el par *username* y *uid*. El atributo *username* es clave foránea de la entidad ***User***. El atributo *uid* es un identificador único que tiene asociado cada cuenta de *Dropbox* y que es proporcionado por el mismo servidor. Por último, se almacenarán dos cadenas, *rsaPublicKey* y *rsaPrivateKey*, que representan el par de claves RSA generados por cada usuario y cuenta de *Dropbox*, siendo este par usado para poder subir y descargar ficheros de dicha cuenta, además de para poder realizar firmas digitales (*Ver 4.6.4.2*). Por tanto, en esta entidad se insertarán registros la primera vez que un usuario se asocie con una cuenta de *Dropbox* específica, generándose en este momento el par de claves

RSA. La próxima vez que ese usuario se vincule a esa cuenta de *Dropbox*, no tendrá que realizar esa generación de claves, sino que sólo tendrá que obtener de la base de datos el par de claves RSA para poder seguir subiendo y descargando ficheros. En la *Figura 7* se pueden observar dos registros de la entidad ***Account***.

username	uid	rsaPublicKey	rsaPrivateKey
alex	295260122	DHRSf0yY0lyy7hpK2G86uP1qjRJyrADiUk5wWy/31b8ACUOKnSDfFtINkBHWhtg0iti3a9Fj9xHh	OApLkn9WcR2J0/FXqtKnOpx+EEuyYT/vYgZUoGpmwnU9XM2/iEY8C0BQnuSlFG5zJ6CqNVih5Rmx
alex	296272265	82G5XZprGASPkB0Zt5DI0dol9duKT2YcyqAHfDy3SoipBLo9oatyKyVhMTTWlUg1PRIo44XcLECJ	javgHZW76NJiL7LK0l+NaSnlUh/e0Djj9b2KSPyBtwrxnqexNh+ZsoHztQDUwNfejFyKm5t0fc6q

Figura 7. Registros de la entidad *Account*

A continuación se tiene la entidad ***UsersSharedFolder***, que es la encargada de almacenar para cualquier cuenta de *Dropbox* vinculada, los usuarios que se encuentran en cada una de las carpetas compartidas de dicha cuenta. Los registros de esta entidad se van actualizando cada vez que se realiza una actualización satisfactoria de los usuarios que pertenecen a una carpeta compartida. Esta entidad es necesaria para que en el caso en el que un servidor con mala intención modifique el fichero en el cual se encuentran los usuarios que pertenecen a cada carpeta compartida, se pueda acceder a la base de datos y recuperar la última versión correcta de dicho fichero. La clave primaria de esta entidad está formada por el trío *username* (clave foránea de ***User***), *uid* (clave foránea de ***Account***) y por *sharedFolder*, que es la ruta del fichero de usuarios de la carpeta compartida. El último atributo es *usersFolders*, que guarda los usuarios que están en la carpeta compartida, en un formato que será detallado en el apartado *4.7.1.2*. En la *Figura 8* se puede observar un registro de la entidad ***UsersSharedFolder***.

username	uid	sharedFolder	usersFolder
alex	295260122	/COMPARTIDA/usuariosCarpeta.txt	MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvGRTfpnh8sawK1UF/03yosq3LQCAd3le

Figura 8. Registro de la entidad *UsersSharedFolder*

La entidad ***RequestSharedFolder***, almacena todas las peticiones de unión o abandono a carpetas compartidas que realiza un usuario vinculado a una cuenta de *Dropbox*. Cada vez que un usuario realiza una petición de unión o de abandono a una carpeta compartida, deberá ser gestionada por el propietario de dicha carpeta. Esta entidad sirve para poder tener un registro de las peticiones que se han realizado, con el objetivo de poder comprobar que han sido gestionadas y además poder volverlas a realizar en caso de que un servidor con mala intención las eliminase. La clave primaria de esta entidad está formada por el cuarteto *username* (clave foránea de ***User***), *uid* (clave foránea de ***Account***), por *sharedFolder*, que es la ruta de la carpeta compartida y por *request*, que se trata de una cadena que indica si es una petición de unión a una carpeta compartida o de abandono. En la *Figura 9* se puede observar un registro de la entidad ***RequestSharedFolder***.

username	uid	sharedFolder	request
alex	295260122	/COMPARTIDA	abandono

Figura 9. Registro de la entidad *RequestSharedFolder*

Por último, se encuentra la entidad ***FilesInPath***, que se encarga de almacenar el nombre de todos los ficheros subidos por un usuario a una cuenta de *Dropbox*. Este registro tiene como principal objetivo ayudar al usuario a comprobar si un servidor malicioso ha eliminado o movido algún fichero que haya subido previamente. La clave primaria de esta entidad está formada por el trío *username* (clave foránea de ***User***), *uid* (clave foránea de ***Account***) y *path*, que es la ruta donde se ha subido el fichero. El último atributo, llamado *files*, es una concatenación de cadenas separadas por el carácter ';', que representa los nombres de los ficheros subidos a una ruta determinada. En la *Figura 10* se puede observar dos registros de la entidad ***FilesInPath***.

username	uid	path	files
alex	296272265	/COMPARTIDA/	usuariosCarpeta.txt
alex	295260122	/COMPARTIDA/	Black-White.jpg;fondo1.jpg

Figura 10. Registros de la entidad *FilesInPath*

En cuanto a las relaciones entre las entidades, destacar que un usuario (***User***) puede tener desde cero hasta N cuentas de *Dropbox* vinculadas (***Account***), y cada cuenta de *Dropbox*, puede tener también desde cero hasta N carpetas compartidas (***UsersSharedFolder***), peticiones a carpetas compartidas (***RequestSharedFolder***) y ficheros en una determinada ruta (***FilesInPath***).

4.5.2. Diagrama gráfico de la aplicación

El diagrama gráfico de la aplicación sirve para mostrar de una manera gráfica y sencilla la navegabilidad de las distintas pantallas que la constituyen, y además en este caso concreto (*Ver Figura 11*), también se puede observar el posible acceso de cada actividad tanto a la base de datos local como al servidor de *Dropbox*.

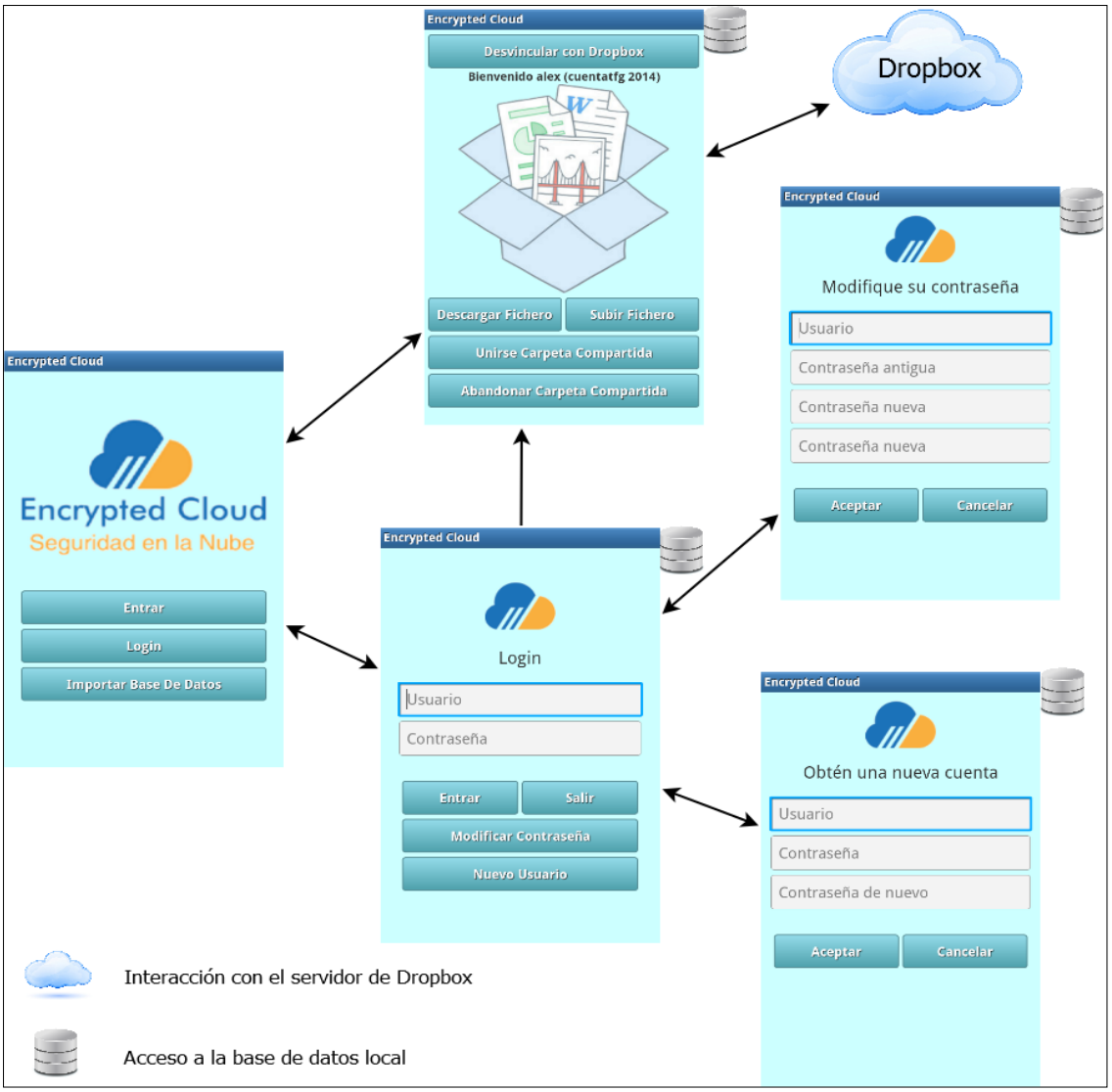


Figura 11. Diseño gráfico

La aplicación está formada por cinco actividades distintas.

La actividad ***Splash*** es la inicial de la aplicación, que contiene una animación que una vez finalizada, dejará visibles tres opciones diferentes. La opción "Entrar" permite a un usuario navegar hacia ***DropboxManager***, siempre y cuando previamente se haya *logueado* al menos una vez. La opción "*Login*" permite navegar hacia la actividad ***Login***, desde la cual podrá *loguearse*, cambiar la contraseña de una cuenta y crear un nuevo usuario. Por último, la opción "Importar Base De Datos" realizará una importación de una base de datos guardada en la *sdcard* del dispositivo móvil, sólo en el caso de que se haya realizado anteriormente una exportación.

La actividad ***Login*** permite a un usuario *loguearse* para poder acceder a la actividad principal de la aplicación, ***DropboxManager***. Para ello, deberá rellenar el campo usuario y contraseña y pulsar sobre el botón "Entrar". También tiene la posibilidad de modificar una contraseña de una cuenta previamente creada, pulsando sobre "Modificar contraseña" o de crear un nuevo usuario en la aplicación, mediante el botón "Nuevo usuario". Si se pulsa sobre el botón "Salir", se volverá a la actividad ***Splash***. Esta actividad accede a la base de datos local para comprobar las credenciales del usuario que se quiere *loguear*.

La actividad ***ModifyPassword*** es la encargada de modificar una contraseña de un usuario que ya está registrado en la aplicación. Para realizar esta modificación, deberá escribir su nombre de usuario, su contraseña antigua y repetir dos veces su contraseña nueva. Una vez rellenados estos campos correctamente, modificará su contraseña pulsando sobre el botón "Aceptar". En cambio, si pulsa sobre el botón "Cancelar", se regresará a la actividad ***Login***. Esta actividad accede a la base de datos local para actualizar los datos del usuario que quiere modificar su contraseña.

La actividad ***Account*** permite a un usuario crearse una nueva cuenta en la aplicación. Se tendrán que rellenar los campos usuario y repetir dos veces la contraseña por cuestiones de seguridad. A continuación, deberá pulsar sobre el botón "Aceptar". Por otro lado, si pulsa sobre el botón "Cancelar", se regresará a la actividad ***Login***. Esta actividad accede a la base de datos local para poder crear un nuevo registro correspondiente a un nuevo usuario.

Por último, se encuentra la actividad principal ***DropboxManager***. Siempre que se navegue a esta actividad, se realizará un *backup* del estado actual de la base de datos local. Se tienen cinco opciones distintas. Con el botón situado en la parte superior, se podrá vincular o desvincular con una cuenta de *Dropbox* previamente creada. Mediante la opción "Descargar Fichero", se tendrá la posibilidad de descargar un fichero cifrado que se encuentre en la cuenta de *Dropbox* que en ese momento está vinculada. Si se pulsa sobre el botón "Subir Fichero", se podrá subir un fichero del dispositivo móvil a cualquier carpeta de la cuenta de *Dropbox* vinculada. Por último, mediante las opciones "Unirse Carpeta Compartida" y "Abandonar Carpeta Compartida", se tendrá la opción de unirse o abandonar una carpeta compartida que esté creada en la cuenta de *Dropbox* con la que se ha vinculado. Esta actividad accede a la base de datos local para poder asociar cuentas de *Dropbox* a los distintos usuarios registrados, además de poder guardar los usuarios que hay en un cierto momento en una carpeta compartida, las peticiones realizadas a carpetas compartidas y los ficheros subidos por una determinada cuenta. También hay que decir que esta actividad es la única que se comunica con el servidor de *Dropbox*.

4.6. Implementación

En esta sección se van a detallar todos aquellos aspectos relacionados con la implementación de la aplicación software desarrollada.

4.6.1. Entorno de desarrollo

Para un nuevo programador, la forma más sencilla de empezar a trabajar en *Android* es descargar el *Bundle ADT*, que contiene los componentes esenciales de *Android* junto con una versión del entorno de desarrollo Eclipse [19].

Por ello, esta ha sido la herramienta seleccionada para desarrollar toda la aplicación, que se encuentra disponible en el siguiente enlace:

http://developer.android.com/sdk/index.html

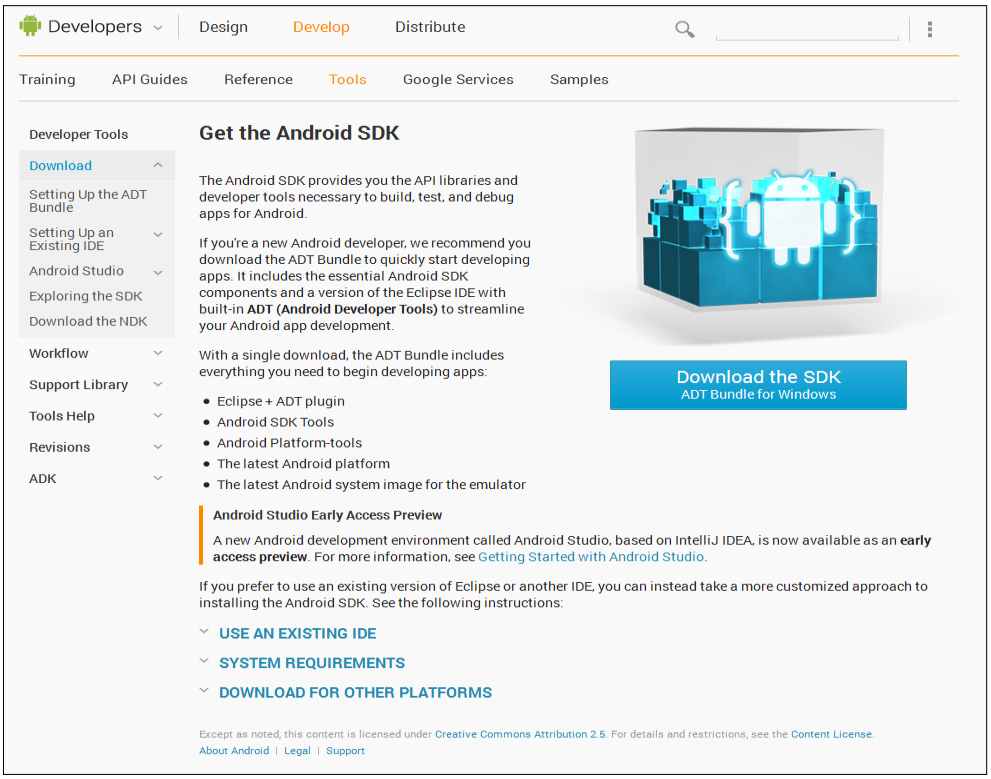


Figura 12. ADT *Bundle*

4.6.2. Herramientas empleadas

En este trabajo, ha sido muy importante la gestión de las referencias consultadas. Al principio, no se manejaban muchas referencias, pero a medida que se fue avanzando, la siguiente herramienta ha sido imprescindible.

Mendeley es un gestor bibliográfico que funciona como un programa de escritorio (para *Windows*, *Mac* y *Linux*) con un *plug-in* para procesador de texto (*Word* de *Microsoft* y *Open Office*) que nos permite recopilar, compartir a través de Internet, gestionar y construir colecciones de referencias bibliográficas [20].

También ha sido muy importante la utilización de un proyecto llamado ***AndroidStringResourceTranslator***, utilizado para poder internacionalizar la aplicación

desarrollada en diferentes idiomas. Gracias al uso de esta herramienta [21], un desarrollador se puede ahorrar el trabajo de ir traduciendo con un traductor todas las palabras empleadas en su aplicación.

Por último, mencionar también la herramienta ***Roboelectric***, que ha sido clave en cuanto a la realización de pruebas unitarias de la aplicación desarrollada se refiere. ***Robolectric*** es un *framework* que simula la interfaz gráfica de un emulador de *Android* permitiéndonos ejecutar test en la propia máquina virtual de Java y ahorrándonos el tiempo de arranque cada vez que queremos ejecutar alguna prueba [22].

4.6.3. Estructura y documentación del código

Con el objetivo de poder separar cada clase o actividad según la función que desempeñan, se ha divido el proyecto en ocho paquetes diferentes.

En *Android* los nombres de los paquetes siguen la convención de poner el dominio en orden inverso. En este caso, los nombres de los paquetes siempre empiezan por:

es.uam.eps.tfg.encryptedcloud

Las ventajas de seguir esta convención son:

- Los nombres de los paquetes son únicos y se evitan colisiones de nombres entre empresas y organizaciones.
- El hecho de que se use el nombre en orden inverso es para que los paquetes dentro de una organización queden bien estructurados. Hay que tener en cuenta que cada '.' se traduce en un directorio. De este modo los directorios se organizan de más general ('es', en este caso) a más particular ('tfg', acrónimo de Trabajo de Fin de Grado en este caso) [19].

A continuación, se van a detallar los ocho paquetes distintos en los que se ha dividido el código de la aplicación.

- **es.uam.eps.tfg.encryptedcloud.activities.** En este paquete se encuentran incluidos todos aquellos elementos que son actividades, que se corresponden con las distintas pantallas de la aplicación.
    - **Account.java.** Actividad encargada de la creación de una nueva cuenta de usuario.
    - **DropboxManager.java**. Actividad principal de la aplicación, encargada de gestionar todas las funcionalidades referentes al uso de *Dropbox*, como son el vincularse o desvincularse de una cuenta, el subir o descargarse un fichero y también la unión y abandono de carpetas compartidas.
    - **Login.java.** Actividad encargada de *loguear* a un usuario previamente registrado.
    - **ModifiyPassword.java.** Actividad encargada de modificar la contraseña de una cuenta existente.
    - **Splash.java.** Actividad inicial de la aplicación.

- **es.uam.eps.tfg.encryptedcloud.asynctasks.** En este paquete se encuentran incluidas todas las tareas asíncronas de la aplicación, que son ejecutadas en un hilo distinto al principal.
    - **DropboxFolderTask.java.** Tarea asíncrona encargada de descargar todos los nombres de carpetas y ficheros de una cuenta de *Dropbox*.
    - **JoinFolderTask.java.** Tarea asíncrona encargada de realizar la primera petición de unión de un usuario a una carpeta compartida de *Dropbox*.
    - **LeaveFolderTask.java.** Tarea asíncrona encargada de realizar la primera petición de abandono de un usuario a una carpeta compartida de *Dropbox*.
    - **KeyStoreTask.java.** Tarea asíncrona encargada de generar un par de claves RSA para una nueva vinculación de una cuenta *Dropbox* a un usuario, o si esa vinculación ya se había hecho, esta tarea será la encargada de recuperar de la base de datos su par de claves RSA.
    - **PrepareFileDownloadTask.java.** Tarea asíncrona encargada de realizar la preparación de descarga de un fichero de una cuenta de *Dropbox*. Esta tarea se encarga, por ejemplo, de comprobar si un usuario pertenece a una carpeta compartida antes de poder descargarse un fichero de la misma.
    - **PrepareFileUploadTask.java.** Tarea asíncrona encargada de realizar la preparación de subida de un fichero de una cuenta de *Dropbox*. Esta tarea se encarga, por ejemplo, de comprobar si un usuario pertenece a una carpeta compartida antes de poder subir un fichero a la misma.
    - **DownloadFileDropboxTask.java.** Tarea asíncrona encargada de descargar un fichero de una cuenta de *Dropbox*.
    - **UploadFileDropboxTask.java.** Tarea asíncrona encargada de subir un fichero a una cuenta de *Dropbox*.
- **es.uam.eps.tfg.encryptedcloud.crypto.** En este paquete se encuentran todas las clases que están relacionadas con el cifrado y descifrado y con el manejo de bytes.
    - **StoreRsaKeyPair.java.** Esta clase se encarga de almacenar el par de claves RSA, que es generado o recuperado cuando un usuario se vincula a una cuenta de *Dropbox*.
    - **Base64.java.** Es una biblioteca perteneciente a *The Android Open Source Project* y proporciona una serie de utilidades para codificar y decodificar de la representación Base64 de datos binarios.
    - **Byte.java.** Clase que proporciona una serie de utilidades para manejar bytes.
    - **Crypto.java.** Clase que contiene todos los métodos criptográficos utilizados en la aplicación.
- **es.uam.eps.tfg.encryptedcloud.database.** En este paquete se encuentra una clase encargada de realizar operaciones con la base de datos.
    - **DatabaseAdapter.java.** Clase encargada de realizar la gestión de la base de datos local.

- **es.uam.eps.tfg.encryptedcloud.dropbox.** En este paquete se encuentran aquellas clases que proporcionan distintas utilidades para integrar la API de *Dropbox* en la aplicación.
    - **DropboxAPIKeys.java.** Clase que proporciona una serie de métodos para garantizar que la API de *Dropbox* ha sido correctamente vinculada a la aplicación.
    - **DropboxUtilities.java**. Clase que proporciona una serie de herramientas relacionadas con *Dropbox*, y que son usadas en otras clases de la aplicación.
- **es.uam.eps.tfg.encryptedcloud.listActivities.** En este paquete se encuentran aquellas actividades que muestran un listado de elementos.
    - **FilePickerActivity.java.** Lista que muestra todos los archivos y carpetas de un dispositivo móvil.
    - **ListFoldersActivity.java.** Lista que muestra todas las carpetas existentes en una cuenta *Dropbox*.
    - **ListFoldersAndFilesActivity.java.** Lista que muestra todos los archivos y carpetas existentes en una cuenta *Dropbox*.
    - **ListFoldersSharedActivity.java.** Lista que muestra todas las carpetas compartidas existentes en una cuenta *Dropbox*.
- **es.uam.eps.tfg.encryptedcloud.preferences.** En este paquete se encuentra una clase encargada de realizar operaciones con las preferencias de la aplicación.
    - **Preferences.java.** Clase encargada de gestionar las preferencias de la aplicación.
- **es.uam.eps.tfg.encryptedcloud.services.** En este paquete se encuentran todos los servicios implementados en la aplicación, encargados de ejecutar cualquier tipo de acción en segundo plano, ya que no requiere interacción directa con el usuario, pero siempre en el hilo principal de la aplicación.
    - **JoinFolderService.java.** Servicio encargado de monitorizar una petición de unión de un usuario a una carpeta compartida.
    - **LeaveFolderService.java.** Servicio encargado de monitorizar una petición de abandono de un usuario de una carpeta compartida.
    - **UpdateSharedFoldersService.java.** Servicio encargado de monitorizar y gestionar las peticiones de unión y abandono realizadas en cada una de las carpetas compartidas de una cuenta de *Dropbox*, así como de actualizar los ficheros de las mismas si fuera necesario.
    - **CheckFilesInFoldersService.java**. Servicio encargado de monitorizar que los ficheros subidos a una cuenta de *Dropbox* no han sido eliminados o movidos.
    - **NewFilesInSharedFolderService.java.** Servicio encargado de comprobar si hay nuevos archivos subidos en una carpeta compartida para poder registrarlos en la base de datos con el objetivo de poder monitorizar su posible eliminación indebida.

En cuanto a la documentación del código, destacar que cada una de las clases anteriores han sido documentadas de forma detallada. Se piensa que la documentación del código es un aspecto sumamente importante, ya que facilita su mantenimiento y ofrece la posibilidad de reutilizar parte del programa en otras aplicaciones, sin necesidad de conocerse de manera profunda el código implementado.

4.6.4. Implementación de las funcionalidades criptográficas

En esta sección se van a detallar todos aquellos aspectos criptográficos usados en la implementación de la aplicación en la plataforma *Android*.

En vista a los requisitos especificados anteriormente, se va a necesitar hacer uso de la cifra simétrica y asimétrica. En la medida que una máxima de la seguridad es no crear algoritmos criptográficos que ya están implementados, se va a hacer uso de los ya desarrollados en el *framework* JCA de Java.

4.6.4.1. Java Cryptographics Architecture (JCA)

Java Cryptographics Architecture (JCA) es un *framework* para criptografía que forma parte de la distribución estándar de la JVM (máquina virtual de Java). Reemplaza (y amplia) el API JCE (*Java Cryptographic Extensions*) [23].

Ofrece un API (*application programming interface*) que permite:

- Generación de claves (claves secretas y pares de claves pública y privada).
- Cifrado simétrico (DES, 3DES, IDEA…).
- Cifrado asimétrico (RSA, DSA, *Diffie-Hellman*, *ElGamal*…).
- Funciones resumen y algoritmos MAC (*Message Authentication Code*).
- Generación y validación de firmas.

La implementación de los distintos algoritmos de cifrado, generación de claves y demás utilidades criptográficas, son ofertadas por paquetes externos denominados *providers*.

- La distribución básica de Java incluye por defecto el *provider* "*SUN*" con implementaciones de los algoritmos más representativos.
- Otros fabricantes ofrecen *providers* adicionales que incluyen nuevos algoritmos o implementaciones alternativas de los ya existentes en el *provider* "*SUN*".

Para usar las clases y métodos del API JCA las aplicaciones tienen que importar, como mínimo, los siguientes paquetes:

```
import java.security.*;
import java.security.interfaces.*;
import java.security.spec.*;

import javax.crypto.*;
import javax.crypto.interfaces.*;
import javax.crypto.spec.*;
```

Hay que destacar que, aunque se ha estudiado y probado otros proveedores criptográficos como *Spongy Castle* [24], al final ha sido suficiente con el proveedor por defecto de Java, "*SUN*", para llevar a cabo la implementación de la aplicación.

Se ha utilizado la cifra simétrica para realizar el cifrado de todos los activos subidos al servidor *cloud*, debido a que es más rápida que la cifra asimétrica. El algoritmo elegido ha sido el AES 256, debido a que es uno de los más seguros y utilizados hoy en día de clave simétrica. En cuanto al modo de operación seleccionado, se ha optado por CBC, al ser más seguro que el modo estándar, el conocido como ECB. Además, también se ha usado este algoritmo para cifrar el par de claves RSA de cada cuenta asociada a cada usuario, a la hora de almacenarla en la base de datos. Si el lector desea entrar en más detalle en este algoritmo, se le remite al *Anexo B. AES 256, CBC: Cipher Block Chaining.*

Por otro lado, también se ha tenido que usar la cifra asimétrica, más concretamente el conocido algoritmo RSA. En la implementación, se ha utilizado tanto para firmar todos los ficheros subidos a una cuenta de *Dropbox*, como para poder distribuir una clave simétrica entre varios usuarios. Si se desea conocer más aspectos sobre el algoritmo criptográfico RSA, diríjase al *Anexo C. RSA.*

En los siguientes apartados se van a detallar la firma digital de ficheros, la distribución de una clave simétrica, las funciones resumen utilizadas y el protocolo SSL, aspectos claves en la aplicación implementada.

4.6.4.2. Firma digital de ficheros

Una firma digital es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la entidad originadora de dicho mensaje, y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador [25].

En la *Figura 13*, se puede ver el proceso de firmado de un documento.

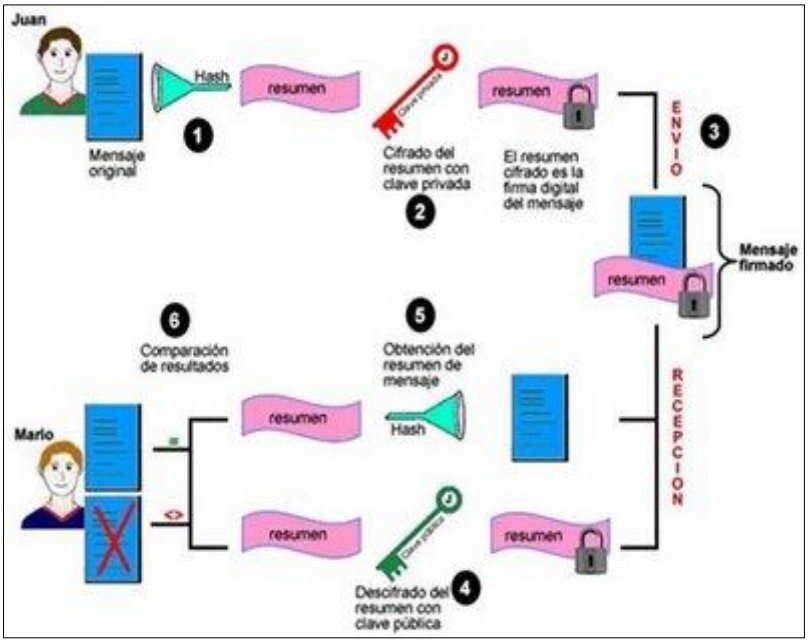


Figura 13. Firma digital [26]

Como se puede observar, lo primero que hace Juan es realizar el *hash* del mensaje original. A continuación, cifra ese resumen con su clave privada. El resumen cifrado es la firma digital del mensaje, enviándolo a Mario junto con el mensaje original.

Mario separa el mensaje original de la firma. Primero calcula el *hash* del mensaje original, obteniendo un resumen. Por otro lado, descifra con la clave pública de Juan la firma digital, obteniendo otro resumen. Por último, compara estos dos resúmenes obtenidos, y en el caso de ser iguales, el firmante será legítimo.

En la aplicación implementada, cada vez que un usuario sube cualquier fichero a una cuenta de *Dropbox*, es firmado digitalmente. Al descargarlo, el usuario podrá comprobar que el firmante es legítimo, y en caso de que no lo fuera, se le notificará pero sin poder recuperar el fichero original no modificado. Sólo podrá ser recuperado el fichero de usuarios en una carpeta compartida, tal y como se va a detallar en el apartado *4.7.6.5*.

4.6.4.3. Distribución de una clave simétrica

El problema principal al que había que enfrentarse en este trabajo era la distribución de una clave criptográfica simétrica entre varios usuarios. Esto es necesario para poder implementar una funcionalidad muy importante ofrecida por *Dropbox*, que es la compartición de archivos en carpetas compartidas. Por lo tanto, había que idear un método para poder subir un archivo cifrado a una carpeta compartida, y que pudiera ser

descifrado por cualquier usuario de dicha carpeta. Como se ha dicho en el apartado *4.6.4.1*, el cifrado del fichero se realiza con una clave simétrica AES de 256 bits, en modo CBC. Para poder realizar la distribución de esta clave con el fin de que cualquier usuario pueda descifrar el fichero, se ha creado y empleado el paquete que se puede visualizar en la *Figura 14*. El formato de este paquete está basado en la envoltura digital, que se puede ver como un contenedor en el cual se encuentra un mensaje cifrado con una clave simétrica, y además se adjunta a este contenido la clave simétrica usada cifrada mediante la clave pública del destinario al cual se le quiere mandar el mensaje.

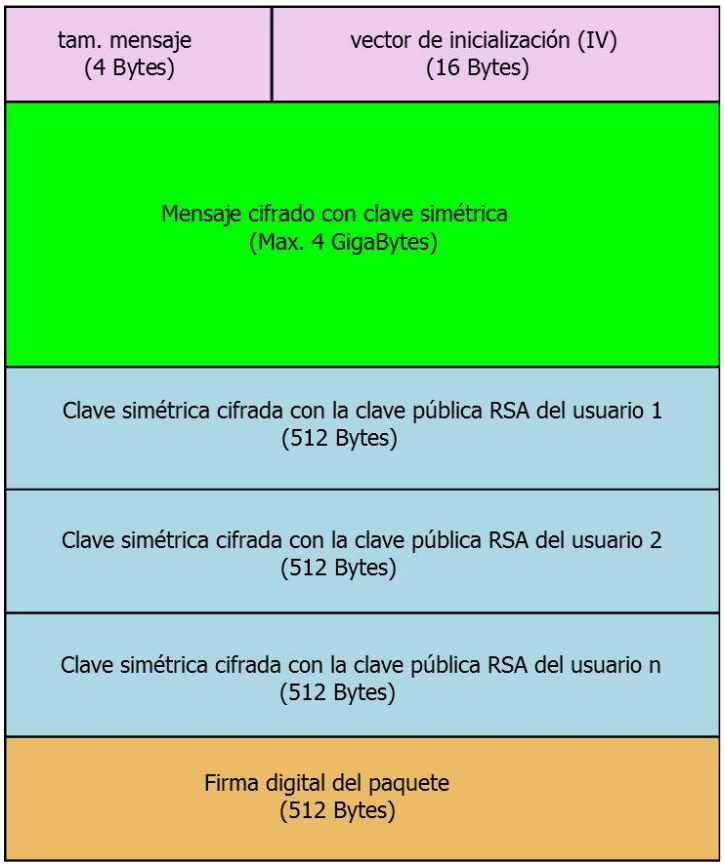


Figura 14. Formato del paquete para intercambio de clave simétrica

El primer campo del paquete tiene un tamaño de 4 Bytes, e indica el tamaño del mensaje cifrado contenido en el paquete. Por lo tanto, este tamaño puede ser como máximo de 4 GigaBytes.

El siguiente campo tiene un tamaño de 16 Bytes, y representa el vector de inicialización usado para realizar el cifrado del mensaje mediante AES 256 CBC.

A continuación, se encuentra el mensaje cifrado mediante el algoritmo criptográfico AES 256 CBC, descrito en el *Anexo B. AES 256, CBC: Cipher Block Chaining.*

Después del mensaje cifrado, se encuentra la clave simétrica cifrada con la clave pública RSA de cada uno de los usuarios con los cuales se quiere compartir el mensaje, y que son los que se encuentran en una carpeta compartida. El orden de cifrado de esta clave simétrica será igual al orden en el cual se encuentran los usuarios unidos en una carpeta

compartida, indicado en un fichero especial que se verá en la sección *4.7.1.2*. Estos campos tienen cada uno un tamaño de 512 bytes para un tamaño de clave RSA de 4096 bits.

Por último, se encuentra la firma digital del contenido del paquete, realizada por el propietario que va a subir un fichero a una carpeta compartida. Esta firma tiene un tamaño de 512 bytes para un tamaño de clave RSA de 4096 bits.

Por tanto, cuando un usuario quiere descifrar un paquete de este tipo, lo primero que tendrá que hacer es saber cuál es su orden dentro de la carpeta compartida, para poder acceder a la posición del paquete donde se encuentra la clave simétrica cifrada con su clave pública, poder obtener esta clave simétrica, y con la misma y el vector de inicialización, llevar a cabo el descifrado del mensaje.

4.6.4.4. Funciones resumen utilizadas

Un *hash* o función resumen es un algoritmo que consigue crear a partir de una entrada (ya sea un texto, una contraseña o un archivo, por ejemplo) una salida alfanumérica de longitud fija que representa un resumen de toda la información que se le ha dado. En otras palabras, a partir de los datos de la entrada se crea una cadena que sólo puede volverse a crear con esos mismos datos [27].

Sabiendo que se puede generar cualquier resumen a partir de cualquier dato se puede realizar la pregunta de si se podrían repetir esos *hash* y la respuesta es que teóricamente sí, que podría haber colisiones, ya que no es fácil tener una función *hash* perfecta.

En la aplicación, se ha usado este tipo de algoritmo criptográfico para guardar la información sensible de un usuario, más concretamente su contraseña de autenticación. Por ello, cuando un usuario se crea una cuenta nueva, se hace un *hash* de su contraseña y se introduce en la base de datos. El hash utilizado es SHA-2 de 256 bits, ya que hasta el momento, no se han encontrado colisiones, como se puede observar en la *Figura 15*.

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Bitwise operations	Collisions found	Example Performance (MiB/s)[6]
MD5 (as reference)		128	128	512	$2^{64}-1$	32	64	and, or, xor, rot	Yes	335
SHA-0		160	160	512	$2^{64}-1$	32	80	and, or, xor, rot	Yes	-
SHA-1		160	160	512	$2^{64}-1$	32	80	and, or, xor, rot	Theoretical attack (2^{61})[9]	192
SHA-2	*SHA-224*	224	256	512	$2^{64}-1$	32	64	and, or, xor, shr, rot	None	139
	SHA-256	256								
	SHA-384	384	512	1024	$2^{128}-1$	64	80	and, or, xor, shr, rot	None	154
	SHA-512	512								
	SHA-512/224	224								
	SHA-512/256	256								
SHA-3	*SHA3-224*	224	1600 (5×5 array of 64-bit words)	1152		64	24	and, xor, not, rot	None	
	SHA3-256	256		1088						
	SHA3-384	384		832						
	SHA3-512	512		576						

Figura 15. Algoritmos *hash* [28]

También se ha empleado este algoritmo criptográfico en la construcción de las firmas digitales realizadas.

4.6.4.5. SSL

SSL (*Secure Socket Layer*) es un protocolo criptográfico empleado para realizar conexiones seguras entre un cliente y un servidor [29].

De forma básica, una conexión usando el protocolo SSL funciona de la siguiente forma:

El cliente y el servidor entran en un proceso de negociación, conocido como *handshake* (apretón de manos). Este proceso sirve para que se establezca varios parámetros para realizar la conexión de forma segura. Una vez terminada la negociación, la conexión segura es establecida.

Usando claves preestablecidas, se codifica y decodifica todo lo que se ha enviado hasta que la conexión se cierre.

Todos los servicios ofrecidos y usados por la API de *Drobpox* en el desarrollo de la aplicación, utilizan conexiones SSL, por lo que aumenta la seguridad de comunicación entre la aplicación y *Dropbox*.

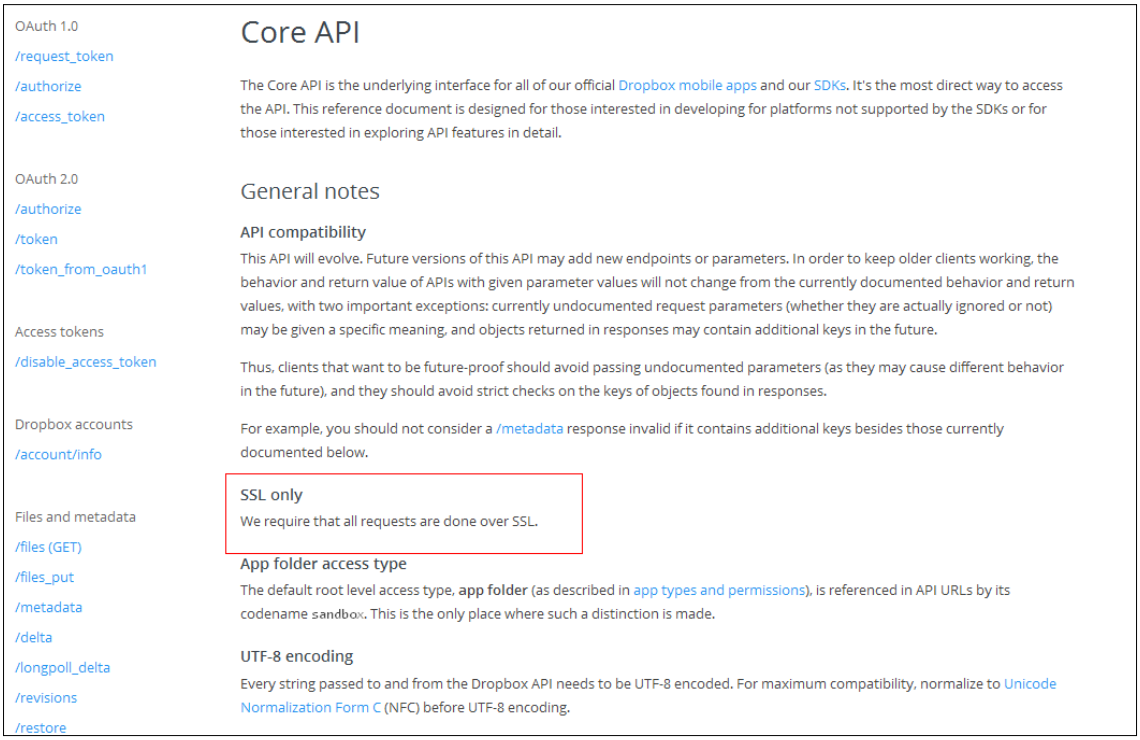


Figura 16. SSL en *Dropbox* API

4.7. Funcionalidad

En este apartado se va a describir muy detalladamente cómo funciona la aplicación desarrollada. Primero se van a especificar una serie de consideraciones que es importante tenerlas presentes para poder comprender determinados aspectos de la aplicación. A continuación, se explicará cada funcionalidad, con el objetivo de que esta sección sirva también como un manual de usuario.

4.7.1. Consideraciones a tener en cuenta

A continuación, se van a exponer una serie de consideraciones que es importante tenerlas en cuenta a la hora de entender la aplicación.

4.7.1.1. Creación y tipos de carpeta en una cuenta de Dropbox de la aplicación

Desde el primer momento, se intentó que la aplicación desarrollada proporcionase todas las funcionalidades que una cuenta de *Dropbox* ofrece a través de su cliente web. Sin embargo, cuando se comenzó el estudio de la API con la que se ha trabajado, se encontró el problema de que en la versión actual no se proporcionaba el servicio de creación de carpetas compartidas, ya que iba a ser incluido en versiones posteriores.

Por tanto, se tomó la decisión de que cualquier cuenta de *Dropbox* que vaya a ser usada en la aplicación, tendrá que crear previamente las carpetas con las que se quiera trabajar, ya que se considera que esta creación de carpetas es algo secundario respecto al problema que se quiere estudiar en este trabajo. La aplicación desarrollada tampoco ofrece la funcionalidad de eliminar ni carpetas ni ficheros.

Para acceder a la información de una carpeta o de un fichero de *Dropbox*, se realiza a través de una función llamada "*metadata*" (*Ver Figura 17*). El problema encontrado ha sido al diferenciar una carpeta no compartida de una compartida, ya que la API no proporciona ningún atributo para distinguirlas.

Figura 17. Atributos de la función "*metadata*" de la API de *Dropbox*

La única forma encontrada de poder hacer esto ha sido a través del icono de la carpeta, ya que son distintos el icono de una carpeta normal y el de una carpeta compartida como se puede observar en la *Figura 18*.

Figura 18. Iconos carpeta no compartida y carpeta compartida

El problema es que según las funcionalidades dadas por la API de *Dropbox*, es imposible identificar que una carpeta que se encuentra dentro de una carpeta compartida también es compartida, ya que estas carpetas internas tienen el mismo icono que las carpetas no compartidas *(Ver Figura 19).*

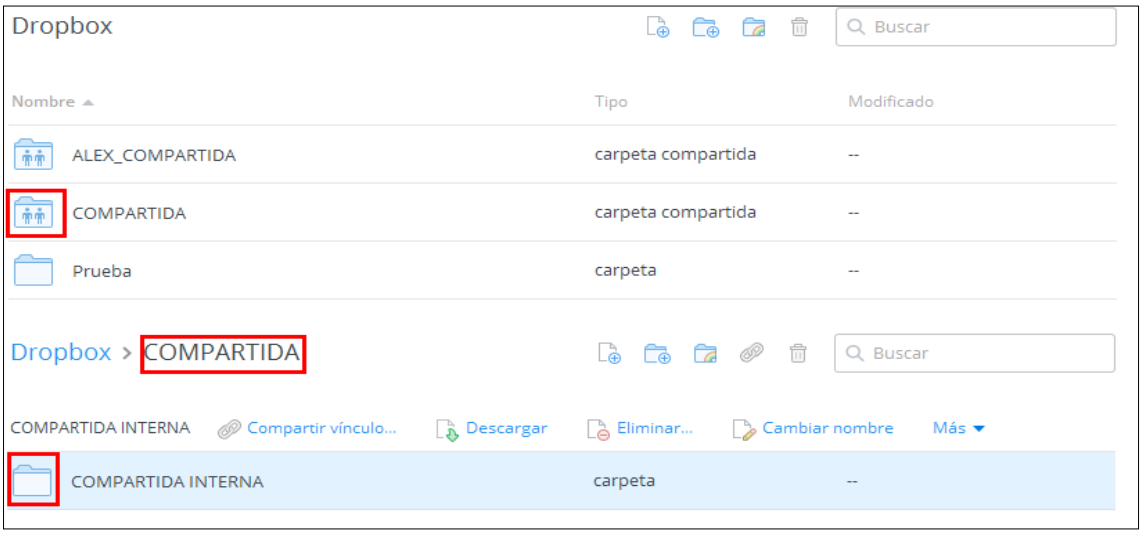


Figura 19. Icono carpeta compartida interna

Una vez realizado este estudio, se tomó la siguiente decisión sobre la naturaleza de las carpetas en una cuenta *Dropbox* dentro de la aplicación desarrollada:

Se van a tener, como ya se ha dicho, dos tipos de carpeta, carpeta no compartida y carpeta compartida.

En el directorio raíz de una cuenta de *Dropbox*, una carpeta podrá ser compartida o no compartida, dependiendo de cómo se haya creado fuera del ámbito de la aplicación.

- Dentro de una carpeta no compartida, **todas** las carpetas creadas serán no compartidas.
- Dentro de una carpeta compartida, todas las carpetas creadas serán compartidas pero independientes entre sí, es decir, cada una tendrá una serie de ficheros y de usuarios que son los que compartirán los recursos. En el siguiente apartado se profundizará más sobre este último aspecto.

4.7.1.2. Gestión de usuarios en una carpeta compartida

Otra carencia que se ha echado en falta de la API de *Dropbox*, pero que era de esperar después de encontrarse que no se podían crear carpetas compartidas, es la funcionalidad de invitar a usuarios a una carpeta compartida *(Ver Figura 20).* Ante este inconveniente encontrado, la aplicación desarrollada asume que para que se puedan compartir archivos

entre varias cuentas, las invitaciones se tendrán que gestionar fuera de la aplicación, ya que no quedan dentro del alcance de la misma.

Figura 20. Problema al compartir carpeta con API de *Dropbox*

No obstante, se ha implementado una funcionalidad propia de unión y abandono de carpetas compartidas. Primero, hay que recordar que a cada usuario que vincula una cuenta de *Dropbox* dentro de la aplicación, se le asocia siempre un par de claves RSA para poder gestionar el cifrado y descifrado de los ficheros de una carpeta compartida. Este par de claves siempre es generado por el usuario, a través de la aplicación cliente. Por otro lado, que una carpeta concreta esté compartida entre dos cuentas, la aplicación lo entiende como una conexión existente entre las dos cuentas pero sin usuarios unidos a esa carpeta, ya que los usuarios que van a unirse y abandonar carpetas compartidas, son los asociados a cada cuenta de *Dropbox* dentro de la aplicación. Para llevar a cabo esta implementación, en una carpeta compartida va a haber un fichero llamado *usuariosCarpeta.txt* en el cual se va a encontrar un listado de los usuarios contenidos en dicha carpeta compartida. Cada usuario es representado por su clave pública RSA, y es separado del resto mediante el carácter ‘;’, como se puede observar en la *Figura 21*.

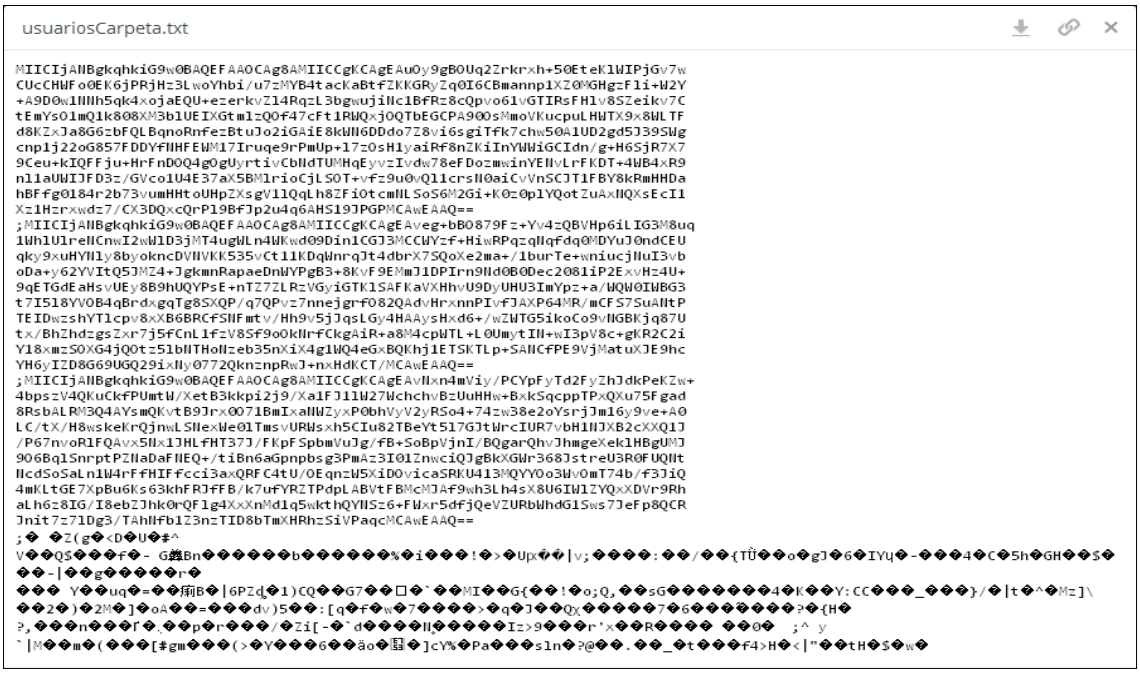


Figura 21. Formato fichero *usuariosCarpeta.txt*

El primer usuario que se una a una carpeta compartida, va a ser el propietario de la misma, y estará en primer lugar en el archivo de usuarios. El fichero *usuariosCarpeta.txt* siempre irá firmado digitalmente por el propietario de la carpeta compartida. El resto de usuarios que se quieran unir o abandonar una carpeta compartida de la que no son propietarios, deberán mandar una petición a ese propietario que deberá procesar. Este comportamiento será descrito en las secciones *4.7.6.3*, *4.7.6.4* y *4.7.6.5*.

4.7.2. Pantalla inicial

La primera pantalla que aparece cuando se inicia la aplicación es la mostrada en la *Figura 22*.

Figura 22. Pantalla inicial

En esta pantalla se le muestran tres opciones diferentes al usuario:

- **Entrar.** Si el usuario pulsa sobre esta opción, navegará directamente a la pantalla principal de la aplicación, siempre y cuando se haya *logueado* previamente al menos una vez.
- ***Login*.** A través de esta opción, el usuario accede a tres funcionalidades distintas, que son el *logueo* para poder entrar en la pantalla principal de la aplicación, la creación de un nuevo usuario o la modificación de la contraseña de una cuenta de usuario existente.
- **Importar Base De Datos.** La aplicación desarrollada, siempre que se navega a la pantalla principal, hace un *backup* de la base de datos local en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil. Cuando se está en la pantalla inicial, se le da la opción al usuario de poder importar estos *backups* de la base de datos local. Esta funcionalidad permite que en el caso de que la aplicación se desinstale y posteriormente se vuelva a instalar, se puedan recuperar todas las claves RSA asociadas a cada usuario y cuenta de *Dropbox*, para poder acceder a los ficheros subidos previamente a dicha cuenta, y evitar de

esta manera una nueva generación de claves, perdiendo el acceso a esos ficheros. Siempre que se realice una importación de una base de datos local, se mostrará mediante un mensaje si esa importación se ha podido realizar o no, ya que se debe haber hecho previamente un *backup* de la base de datos.

4.7.3. Login

Como se ha dicho en el apartado anterior, mediante la opción *Login* del menú inicial se puede acceder al *logueo* de un usuario que ya tiene una cuenta creada en la aplicación, como se puede ver en la *Figura 23* .

Figura 23. *Login*

En esta actividad, el usuario podrá realizar tres acciones distintas:

- *Loguearse*, introduciendo sus credenciales y pulsando sobre el botón "Entrar". En el caso de que este proceso tenga éxito, se navegará a la actividad principal de la aplicación, realizando una exportación del estado actual de la base de datos local. En caso contrario, se le informará del error al usuario.
- Modificar una contraseña de una cuenta ya creada, mediante la opción "Modificar Contraseña".
- Crear un nuevo usuario en la aplicación, mediante el botón "Nuevo Usuario".

4.7.4. Creación de un nuevo usuario

Para poder usar la aplicación desarrollada, es imprescindible tener una cuenta de usuario registrada en la aplicación. A esta funcionalidad se accede a través del botón "Nuevo Usuario" de la actividad *Login*, como ya se ha comentado en el apartado anterior.

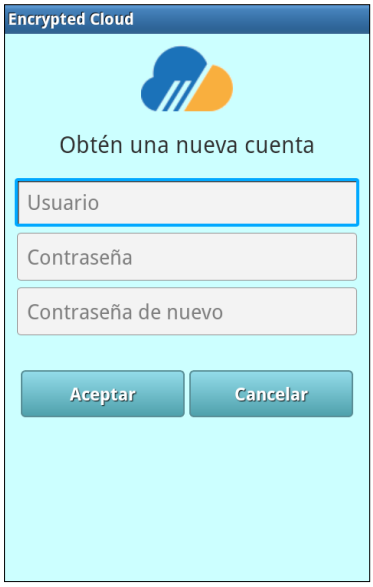


Figura 24. Creación de una cuenta de usuario

Para crear una nueva cuenta, simplemente habrá que especificar un nombre de usuario y repetir dos veces la contraseña que se quiere establecer. Si no hay ningún usuario ya creado con ese mismo nombre, el registro será satisfactorio, se guardará en la base de datos en la tabla ***User***, en el formato explicado en el apartado referente al diseño de la base de datos y se volverá a la actividad de ***Login*** para poder usar las credenciales que se acaban de crear. En caso de que el registro sea erróneo, se le informará debidamente al usuario.

4.7.5. Modificación de la contraseña de una cuenta

Por razones de comodidad, la actividad proporciona al usuario la funcionalidad de poder modificar una contraseña de una cuenta de usuario previamente creada. Evidentemente, el usuario deberá recordar la contraseña actual de dicha cuenta para poder cambiarla.

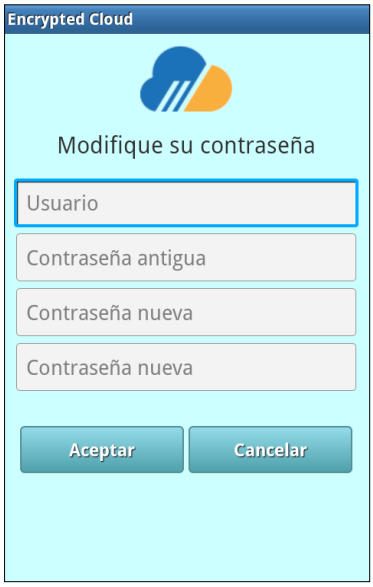


Figura 25. Modificación de la contraseña de una cuenta de usuario

Para poder modificar una contraseña, se deberá proporcionar el nombre de usuario, la contraseña antigua y repetir dos veces la contraseña nueva por la que se quiere sustituir. Si se proporcionan estos datos de manera correcta, la contraseña será modificada y se volverá a la actividad de ***Login***, con el objetivo de poder utilizar estas nuevas credenciales. En caso en el que los datos introducidos sean incorrectos, se le informará debidamente al usuario.

4.7.6. Dropbox Manager

La actividad ***DropboxManager*** es considerada como el corazón de la aplicación. Es la encargada de gestionar todas las funcionalidades que están relacionadas con *Dropbox* y con la aplicación. Por tanto, es la encargada de vincular y desvincular cuentas de *Dropbox*, de realizar la unión o el abandono de usuarios a carpetas compartidas así como de actualizarlas, y también de subir y descargar archivos cifrados comprobando su integridad. Todas estas funcionalidades son las que van a ser descritas en los siguientes puntos.

4.7.6.1. Vincularse a una cuenta de Dropbox

Una vez que un usuario accede a la actividad principal ***DropboxManager***, ya sea mediante la opción "Entrar" de la pantalla inicial o a través de un nuevo *logueo*, se encontrará con una interfaz como la mostrada en la *Figura 26*.

Figura 26. Pantalla inicial *DropboxManager*

Por tanto, lo que el usuario deberá hacer si quiere utilizar todas las funcionalidades que proporciona la aplicación es vincular una cuenta de *Dropbox* ya creada mediante el botón "Vincular con *Dropbox*".

Una vez pulsado este botón, se abrirá el navegador del dispositivo (*Ver Figura 27*), en el cual se mostrará una solicitud para introducir las credenciales de una cuenta de *Dropbox*, como paso previo a la realización de la vinculación.

Figura 27. Inicio de sesión en *Dropbox*

A continuación, se navegará a una pantalla en la cual se pide que la cuenta de *Dropbox* dé permisos para que la aplicación *EncryptedCloud* API pueda tener acceso a todos los archivos y carpetas de la cuenta de *Dropbox*.

Figura 28. Permitir acceso *EncryptedCloud* API

Si se pulsa el botón "Permitir", el control vuelve a la aplicación, la cual lanza una tarea asíncrona encargada de comprobar si el usuario *logueado* ya había realizado una vinculación previa a la misma cuenta de *Dropbox*:

- En caso negativo, la tarea asíncrona primero generará un par de claves RSA para ese usuario vinculado a esa cuenta de *Dropbox*. Una vez generado ese par de claves, se creará un registro nuevo en la base de datos local, en la tabla ***Account***. Se guardará el nombre de usuario, el identificador único de la cuenta de *Dropbox*, y el par de claves RSA cifrado mediante AES 256 CBC.

- En caso afirmativo, la tarea asíncrona únicamente deberá acceder a la base de datos local, e indexar la tabla ***Account*** por los campos que hacen referencia al nombre de usuario que se encuentra *logueado* y al identificador único de la cuenta de *Dropbox* vinculada. De esta forma, se recupera el par de claves RSA asociados, pero se encuentra cifrado. Hay que tener en cuenta que la clave usada para realizar este cifrado mediante AES 256 CBC, es generada desde una semilla creada a partir de la concatenación del nombre de usuario y del identificador de la cuenta de *Dropbox*. Por ello, se puede recuperar sin ningún problema esa clave de cifrado, y posteriormente descifrar el par de claves RSA para poder trabajar con el mismo.

En la *Figura 29* se puede observar el estado de la aplicación mientras se está realizando el proceso anteriormente descrito.

Figura 29. Carga de los datos del usuario

Por último, una vez terminado el proceso de gestión de claves, se mostrará la actividad como se puede observar en la *Figura 30*, pudiendo acceder a todas las funcionalidades de la misma, dándose por finalizado el proceso de vinculación.

Figura 30. Cuenta de *Dropbox* vinculada a la aplicación

4.7.6.2. Desvincularse de una cuenta de Drobpox

En cualquier instante en el cual un usuario se encuentre en la actividad ***DropboxManager*** y desee desvincularse de la cuenta de *Dropbox* actual, únicamente deberá pulsar sobre el botón "Desvincular con *Dropbox*", situado en la parte superior de la *Figura 31*.

Figura 31. Desvincularse de una cuenta de *Dropbox*

Una vez pulsado ese botón, se deshabilitarán todas las funcionalidades que se encuentran en la parte inferior de esta actividad, quedándose con la apariencia que muestra la *Figura 32*.

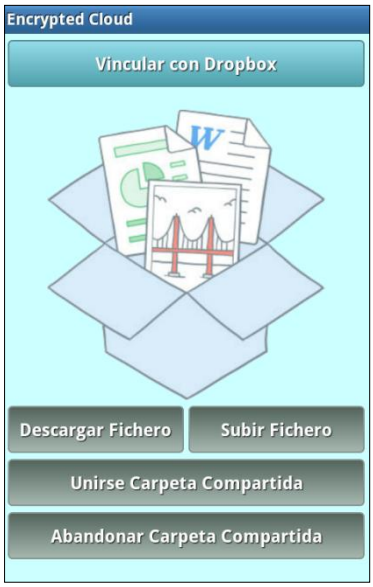


Figura 32. Cuenta de *Dropbox* desvinculada

4.7.6.3. Unirse a una carpeta compartida

Se va a explicar antes la funcionalidad de unirse y abandonar una carpeta compartida que la de subir y descargar ficheros porque éstas últimas dependen en cierta medida de las primeras.

Para poder unirse a una carpeta compartida, lo que primero se deberá hacer es pulsar sobre el botón "Unirse Carpeta Compartida" mostrado en la *Figura 33*.

Figura 33. Unirse a una carpeta compartida

Una vez realizado esto, se cargará en un explorador de ficheros todas las carpetas compartidas que estén ya creadas en la cuenta de *Dropbox* vinculada. En este ejemplo mostrado en la *Figura 34*, sólo hay una carpeta compartida llamada "COMPARTIDA".

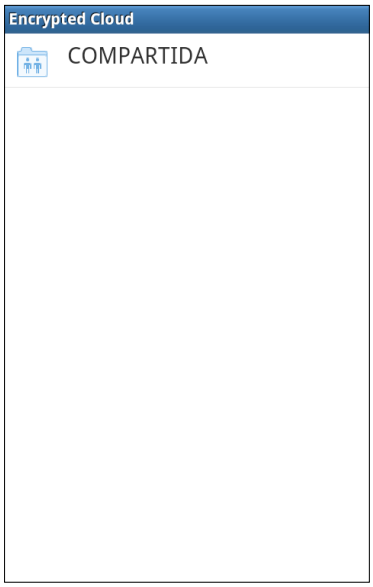


Figura 34. Explorador unión carpetas compartidas

Cuando se pulsa sobre una de las carpetas mostradas, se podrán observar dos comportamientos. Si la carpeta pulsada no tiene carpetas internas, se realizará la petición de unión a dicha carpeta seleccionada. En caso contrario, se mostrarán las carpetas compartidas contenidas en su interior, con una opción llamada "Unirse a esta carpeta", para poder realizar la unión a la carpeta en la cual se encuentra el usuario en ese momento.

Figura 35. Explorador carpeta interna unión carpetas compartidas

De una forma u otra, una vez que se haya seleccionado una carpeta, se procederá a realizar la gestión de unirse a una carpeta compartida, como se puede observar en la *Figura 36*.

Figura 36. Uniéndose a una carpeta compartida

En el proceso de unión a una carpeta compartida, hay que distinguir dos escenarios:

- Si el usuario que está realizando la petición de unirse a la carpeta compartida es el primero que va a haber en esa carpeta, es decir, no hay ningún usuario en dicha carpeta excepto él que se va a unir, se va a convertir de forma automática en el propietario de la carpeta compartida, y se va a poder unir sin ningún problema. Esta unión va a consistir en actualizar o crear el fichero *usuariosCarpeta.txt* si no existe en la carpeta compartida, añadiendo su clave pública RSA seguida de un carácter delimitador ‘;’ y firmando digitalmente dicho fichero. Una vez realizado este proceso, se le informará al usuario que se ha unido correctamente a la carpeta compartida, como es mostrado en la *Figura 37*.

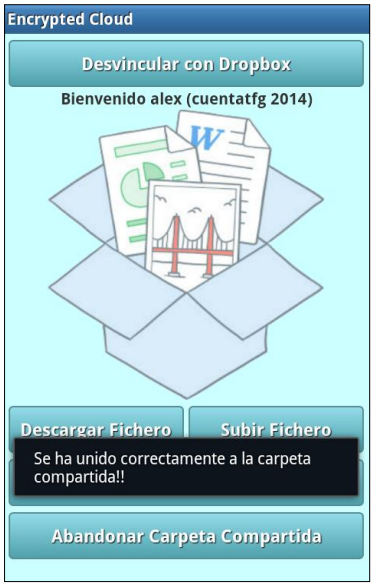


Figura 37. Propietario unido a una carpeta compartida

- En caso contrario, si el usuario que se va a unir a la carpeta compartida no es el primero, es decir, ya existen usuarios registrados en dicha carpeta ya que se encuentran sus claves públicas RSA en el fichero *usuariosCarpeta.txt*, lo que va a hacer es mandar una petición de unión al propietario de dicha carpeta compartida. Esta petición consistirá en un fichero cuyo nombre será la concatenación del nombre de usuario y el identificador de la cuenta de *Dropbox* vinculada, y cuyo contenido será su clave pública RSA concatenada con un identificador (para poder distinguir si se trata de una petición de unión o abandono) firmados digitalmente. Esta petición se subirá a una carpeta dentro de la carpeta compartida a la que se quiere unir, llamada *peticionesUnirseCarpeta*. Además esta petición es registrada en la base de datos local, concretamente en la entidad ***RequestSharedFolder***. Se tuvo en cuenta que un servidor con mala intención pudiera eliminar dichas peticiones, y por tanto no quedaría constancia de que un usuario ha intentado unirse a una carpeta compartida de la que no es propietario. Para resolver este problema, se implementó un servicio que se inicia cuando se entra a la actividad ***DropboxManager***, encargado de monitorizar si dicha petición ha sido gestionada por el propietario. Este servicio comprueba cada 30 segundos si todas las peticiones de unión contenidas en la entidad ***RequestSharedFolder*** de la base de datos local han sido gestionadas correctamente, es decir, si el usuario se encuentra ya incluido en las carpetas compartidas correspondientes, comprobando si su clave pública está en el fichero *usuariosCarpeta.txt*. En el caso de que aún no esté, revisará que las peticiones de unirse que hizo no han sido borradas, y en caso de que lo fueran, volvería a subir cada petición eliminada hasta que por fin estuviera unido a dichas carpetas compartidas. De una forma u otra, una vez realizada una petición de unión a una carpeta compartida, se le informará al usuario mediante un mensaje, como se muestra en la *Figura 38*.

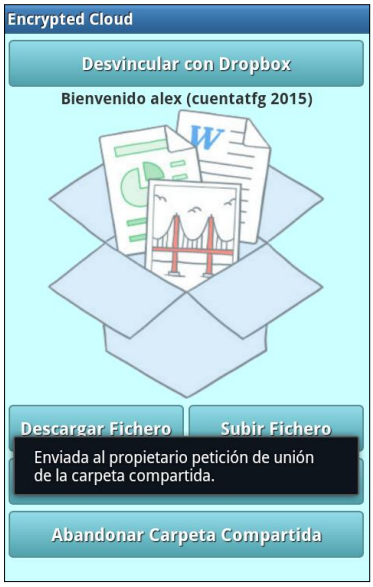


Figura 38. Petición de unión a carpeta compartida

También se puede dar el caso en el que un usuario se intente unir dos veces a una carpeta compartida. Esto no se permitirá, informándole al usuario del error.

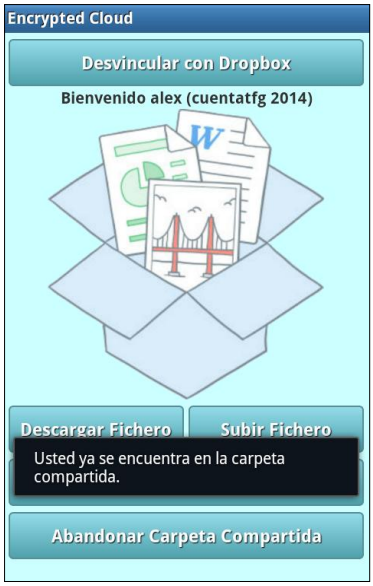


Figura 39. Error al unirse dos veces un usuario a una carpeta compartida

Finalmente, si un usuario intenta unirse a una carpeta compartida cuando ésta se está actualizando, se le notificará que en ese momento no es posible acceder a esa carpeta, como se puede ver en la *Figura 40*. La actualización de carpetas compartidas y la gestión de peticiones por parte del propietario serán detalladas en la sección *4.7.6.5*.

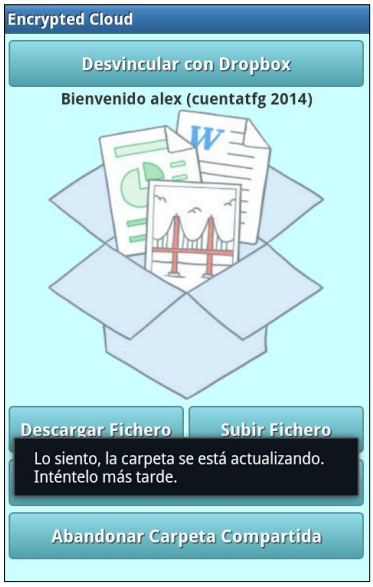


Figura 40. Error al unirse a carpeta compartida actualizándose

4.7.6.4. Abandonar una carpeta compartida

Para poder abandonar una carpeta compartida, lo que primero se deberá hacer es pulsar sobre el botón "Abandonar Carpeta Compartida" mostrado en la *Figura 41*.

Figura 41. Abandonar una carpeta compartida

Una vez realizado esto, se cargará en un explorador de ficheros todas las carpetas compartidas que estén ya creadas en la cuenta de *Dropbox* vinculada. En este ejemplo mostrado en la *Figura 42*, sólo hay una carpeta compartida llamada "COMPARTIDA".

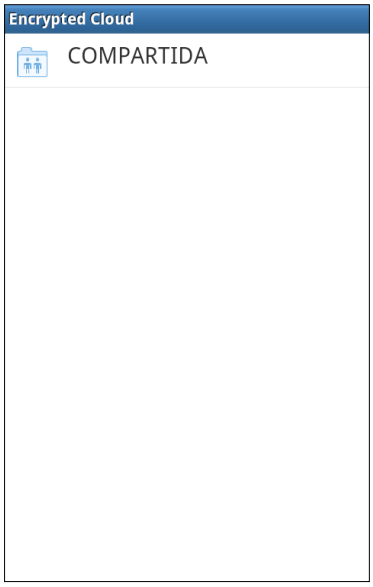


Figura 42. Explorador abandono carpetas compartidas

Cuando se pulsa sobre una de las carpetas mostradas, se podrán observar dos comportamientos. Si la carpeta pulsada no tiene carpetas internas, se realizará la petición de abandono de dicha carpeta seleccionada. En caso contrario, se mostrarán las carpetas compartidas contenidas en su interior, con una opción llamada "Abandonar esta carpeta", para poder realizar el abandono de la carpeta en la cual se encuentra el usuario en ese momento.

Figura 43. Explorador carpeta interna abandono carpetas compartidas

De una forma u otra, una vez que se haya seleccionado una carpeta, se procederá a realizar la gestión de abandonar una carpeta compartida, como se puede observar en la *Figura 44*.

Figura 44. Abandonando una carpeta compartida

En el proceso de abandonar una carpeta compartida, hay que distinguir dos escenarios:

- Si el usuario que está realizando la petición de abandonar la carpeta compartida es el propietario, es decir, su clave pública RSA es la situada en primer lugar del fichero *usuariosCarpeta.txt*, porque fue el primero en unirse, lo único que hará es eliminar todos los ficheros de esa carpeta, ya que se ha tomado el enfoque de que cuando un propietario abandona una carpeta compartida, se reinicia dicha carpeta para realizar en el futuro un nuevo uso de ella. Además, eliminará los ficheros registrados en la base de datos local para esa carpeta compartida, en la entidad ***FilesInPath***. Se le notificará al usuario de que ha abandonado la carpeta compartida, como se puede observar en la *Figura 45*.

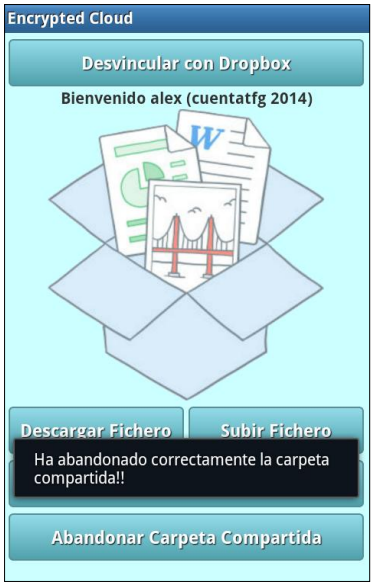


Figura 45. Propietario abandona carpeta compartida

- En caso contrario, si el usuario que va abandonar la carpeta compartida no es el propietario, es decir, su clave pública RSA no se encuentra en primer lugar del fichero *usuariosCarpeta.txt*, lo que va a hacer es mandar una petición de abandono al propietario de dicha carpeta compartida. Esta petición consistirá en un fichero cuyo nombre será la concatenación del nombre de usuario y el identificador de la cuenta de *Dropbox* vinculada, y cuyo contenido será su clave pública RSA concatenada con un identificador (para poder distinguir si se trata de una petición de unión o abandono) firmados digitalmente. Esta petición se subirá a una carpeta dentro de la carpeta compartida que quiere abandonar, llamada *peticionesAbandonoCarpeta*. Además esta petición es registrada en la base de datos local, concretamente en la entidad ***RequestSharedFolder***. Se tuvo en cuenta que un servidor con mala intención pudiera eliminar dichas peticiones, y por tanto no quedaría constancia de que un usuario ha intentado abandonar una carpeta compartida de la que no es propietario. Para resolver este problema, se implementó un servicio que monitoriza si dicha petición ha sido gestionada por el propietario. Este servicio comprueba cada 30 segundos si todas las peticiones de abandono contenidas en la entidad ***RequestSharedFolder*** de la base de datos local han sido gestionadas correctamente, es decir, si el usuario no se encuentra ya incluido en las carpetas compartidas correspondientes, comprobando si su clave pública ya no está en el fichero *usuariosCarpeta.txt*. En el caso de que aún esté, revisará que las peticiones de abandono que hizo no han sido borradas, y en caso de que lo fueran, volvería a subir cada petición eliminada hasta que por fin hubiera abandonado dichas carpetas compartidas. De una forma u otra, una vez realizada una petición de abandono de una carpeta compartida, se le informará al usuario mediante un mensaje como el mostrado en la *Figura 46*.

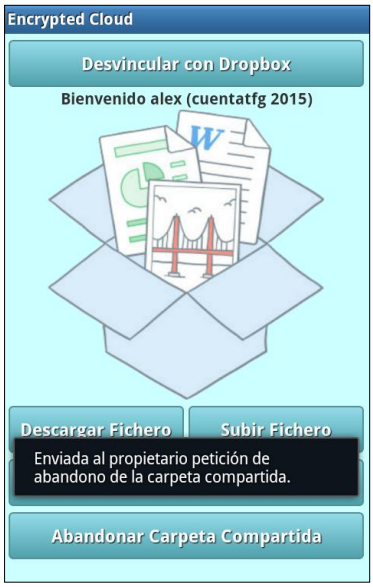


Figura 46. Petición de abandono de carpeta compartida

También se puede dar el caso en el que un usuario intente abandonar dos veces una carpeta compartida. Esto no se permitirá, informándole al usuario del error, como se puede observar en la *Figura 47*.

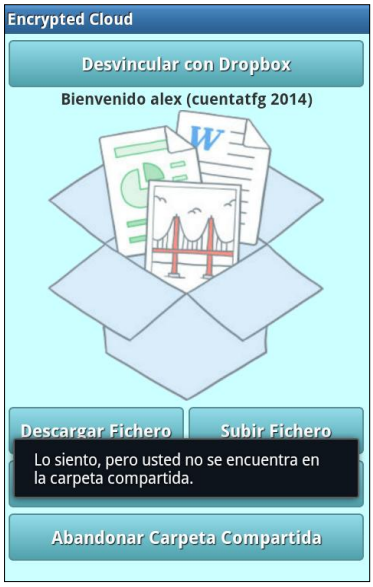


Figura 47. Error al abandonar dos veces un usuario una carpeta compartida

Finalmente, si un usuario intenta abandonar una carpeta compartida cuando ésta se está actualizando, se le notificará que en ese momento no es posible acceder a esa carpeta, como se puede ver en la *Figura 48*.

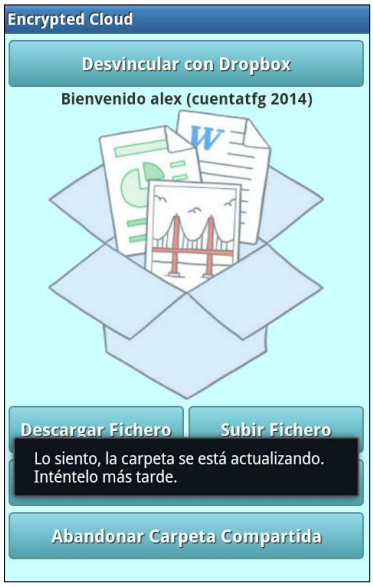


Figura 48. Error al abandonar carpeta compartida actualizándose

4.7.6.5. Actualización de una carpeta compartida

Cada vez que un usuario vincula satisfactoriamente una cuenta de *Dropbox*, se lanza un servicio crucial en la funcionalidad de la aplicación desarrollada.

Este servicio se encarga de recorrer todas las carpetas compartidas que están creadas en esa cuenta de *Dropbox*. Por cada carpeta compartida, revisa si él es el propietario. En caso en el que sí lo sea, observa si dicha carpeta tiene peticiones mandadas por otros usuarios tanto de abandono como de unión.

Como ya se ha dicho antes, el contenido de estas peticiones son la clave pública RSA concatenada con un identificador de petición, firmados digitalmente por el usuario que ha realizado la petición. Por tanto, lo primero que hace el propietario es procesar cada petición contenida en la carpeta compartida, es decir, comprueba que la firma digital es correcta.

Una vez realizado ese proceso de verificación, el siguiente paso es obtener el contenido actual del fichero *usuariosCarpeta.txt* de la carpeta compartida, y actualizarlo añadiendo y eliminando usuarios, según los tipos de peticiones procesados. Si se hubiera hecho una modificación de este fichero por parte de un tercero, el propietario podría recuperar de la base de datos local la última versión de ese fichero.

Finalmente, cuando se tiene el fichero de usuarios actualizado, el propietario se descarga cada uno de los ficheros contenidos en la carpeta, los descifra, y los vuelve a cifrar con una nueva clave AES 256, teniendo en cuenta los nuevos usuarios contenidos en la carpeta compartida. Cada vez que cifra de nuevo un fichero, lo vuelve a subir a la carpeta compartida. Mediante este proceso, se asegura una buena gestión de la revocación y de la unión de usuarios, ya que los usuarios revocados no pueden acceder al contenido de los ficheros de la carpeta compartida, y los nuevos usuarios tienen acceso completo a los mismos una vez que se ha completado el proceso de actualización de la carpeta. Además, siempre que el propietario se encuentre en la carpeta cualquier fichero que no esté debidamente firmado, lo eliminará automáticamente.

Hay que destacar que cuando el propietario está actualizando cualquier contenido de una carpeta compartida, sube a dicha carpeta un fichero llamado *actualizandoCarpeta.txt*, cuyo contenido es una cadena no relevante, firmada por dicho propietario. Este fichero es necesario para poder indicar al resto de usuarios que mientras que dicho fichero se encuentra en una carpeta compartida, ésta no estará disponible, y por tanto no se podrán ni subir ni descargar ficheros de la misma, ni realizar peticiones de unión o abandono. Al terminar la actualización, el propietario siempre comprueba que ese fichero de actualización sigue existiendo y que no ha sido modificado. Finalmente lo termina eliminando. Siempre que se actualiza el fichero de usuarios de una carpeta compartida, se guarda su contenido en la base de datos local, en la entidad ***UsersSharedFolder***, con el fin de poder recuperar ese contenido en el caso en el que el fichero de usuarios actual de una carpeta compartida fuese modificado o eliminado.

Esta manera de llevar a cabo la gestión de las peticiones tiene el problema de que, cuando un usuario realiza una petición a un propietario de una carpeta compartida, tiene que esperar a que éste último la gestione para poder acceder o dejar de acceder a ella. No obstante, se cree que es necesario que el propietario sea el encargado de realizar esta gestión, para poder verificar y tener la seguridad de que las peticiones realizadas a su carpeta compartida son correctas.

4.7.6.6. Subir un fichero

Lo primero que se deberá hacer para poder subir un fichero a una cuenta de *Dropbox* es pulsar sobre el botón "Subir Fichero" que se encuentra en la actividad ***DropboxManager***, resaltado en la *Figura 49*.

Figura 49. Botón Subir Fichero en *DropboxManager*

A continuación, se desplegará un explorador de ficheros, mostrando todas las carpetas y ficheros contenidos en el dispositivo móvil en el cual se está desplegando la aplicación. Se deberá elegir cualquiera de los ficheros mostrados para subirlo a una carpeta de la cuenta de *Dropbox*. En el ejemplo de la *Figura 50*, se selecciona el último fichero.

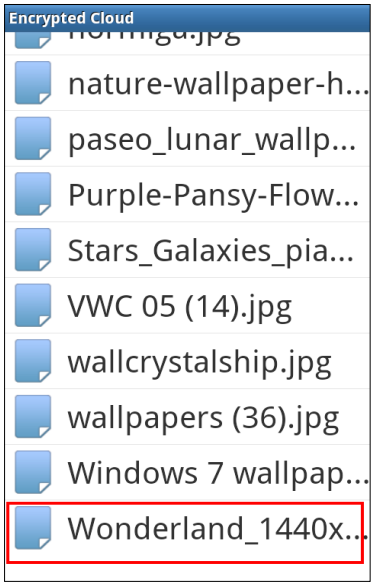


Figura 50. Explorador de ficheros del dispositivo móvil

Una vez seleccionado el fichero que se desea subir, se mostrará otro explorador de carpetas, en el cual se van a listar todas las carpetas que hay disponibles en la cuenta de *Dropbox* que esté vinculada.

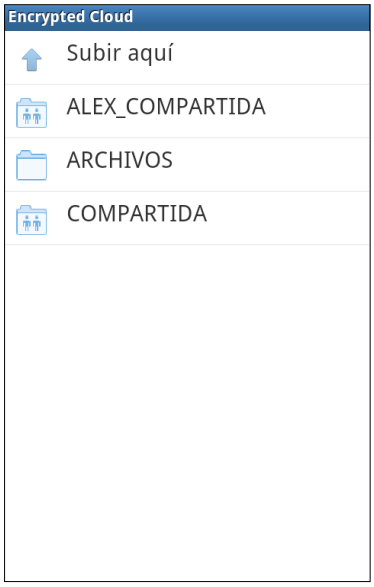


Figura 51. Explorador de carpetas *Dropbox* al subir fichero

Este explorador mostrará todas las carpetas ya sean compartidas o no, y en cada nivel se tendrá una opción llamada "Subir aquí", que dará la opción de subir un fichero en la ruta actual en la que se encuentra el usuario. También se puede navegar en las carpetas contenidas dentro de otras carpetas, como se puede observar en la *Figura 52* el contenido de la carpeta "COMPARTIDA".

Figura 52. Explorador de carpetas interno *Dropbox* al subir fichero

Una vez seleccionada la carpeta en la cual se quiere subir el fichero también previamente seleccionado, se procederá a la preparación del archivo para su subida, como se puede ver en la *Figura 53*.

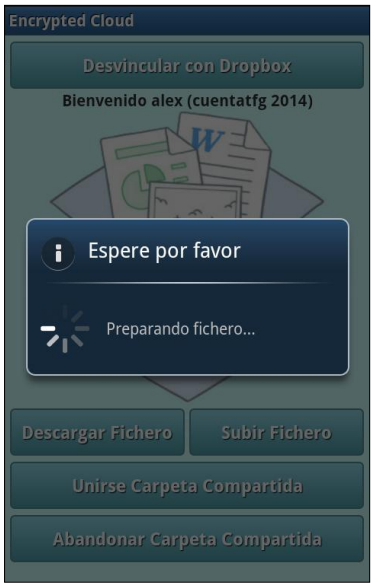


Figura 53. Preparación para subir un fichero a *Dropbox*

Esta preparación del fichero que va a ser subido se tratará de forma distinta dependiendo del tipo de carpeta seleccionado:

- Si se ha seleccionado una carpeta no compartida, el fichero elegido será cifrado mediante AES 256 CBC, generando la clave de cifrado a partir de la semilla creada de la concatenación del nombre de usuario *logueado* en la aplicación y del identificador único de la cuenta de *Dropbox* vinculada. Una vez realizado este cifrado, se realizará su firma digital y se subirá el archivo al destino elegido. El fichero subido se registrará en la base de datos local, en la entidad ***FilesInPath***.
- En cambio, si se ha seleccionado una carpeta compartida, el fichero también se cifrará mediante AES 256 CBC, pero se formará un paquete como el explicado en la sección *4.6.4.3*, encargado de poder distribuir la clave de cifrado del archivo entre los usuarios de la carpeta compartida. Una vez formado este tipo de paquete, será escrito en un fichero, se realizará su firma digital y será subido al destino elegido. Este archivo sólo se registrará en la base de datos local en el caso de que sea el propietario de la carpeta compartida. En caso contrario, no se realizará ningún registro, ya que se dispone de un servicio encargado de monitorizar y registrar los nuevos ficheros subidos a una carpeta compartida en la cual el usuario actual es su propietario.

Una vez preparado el archivo, se procederá a la subida del mismo, indicando al usuario el progreso llevado en todo momento, y finalmente, se mostrará un mensaje notificando que el fichero se ha subido correctamente como se puede ver en la *Figura 54* y en la *Figura 55*.

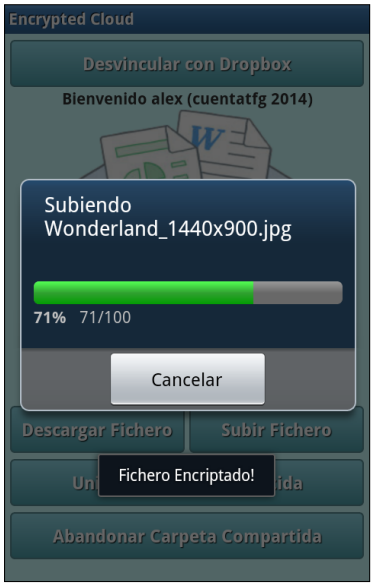


Figura 54. Progreso de la subida de un fichero a *Dropbox*

Figura 55. Fichero subido a *Dropbox* correctamente

Se puede observar en la *Figura 56* y en la *Figura 57* que el fichero es subido sin problemas a la ruta seleccionada, y además que a través de por ejemplo del acceso web a la cuenta de *Dropbox*, no es posible visualizar su contenido.

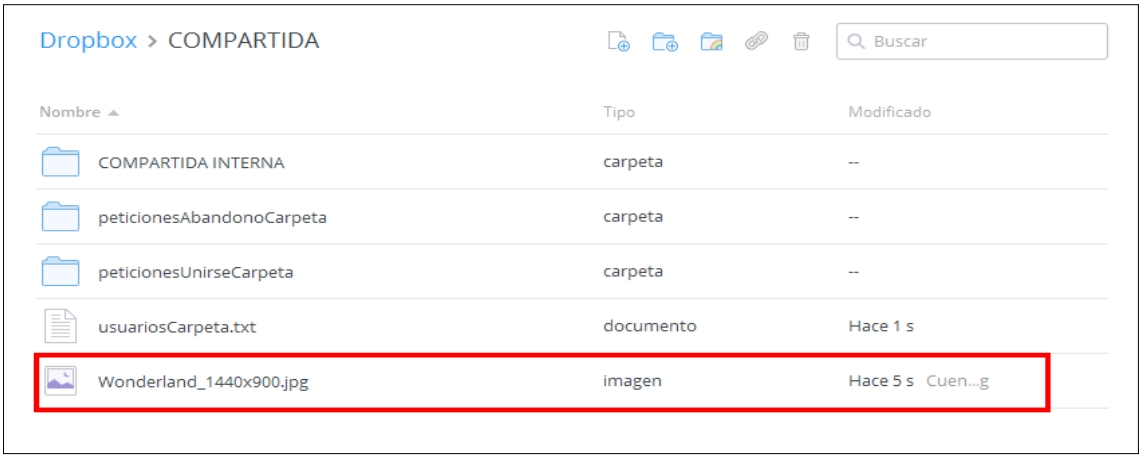


Figura 56. Fichero subido y contenido en carpeta de *Dropbox*

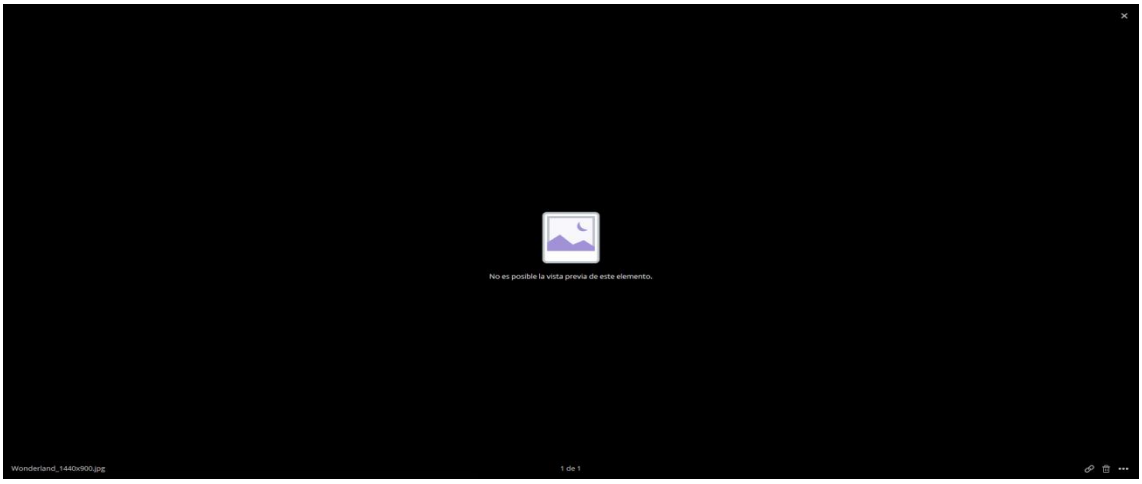


Figura 57. Fichero cifrado en *Dropbox* no puede visualizarse

Hay que tener presente que si un usuario intenta subir un fichero a una carpeta compartida a la cual no pertenece, se le notificará impidiendo que realice esa subida.

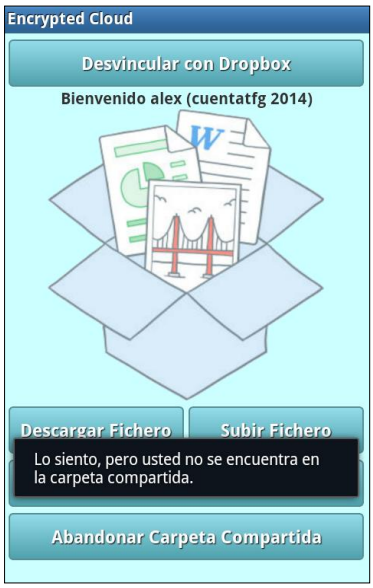


Figura 58. Error al subir fichero a carpeta compartida

Finalmente, si un usuario intenta subir un archivo a una carpeta compartida cuando ésta se está actualizando, se le notificará que en ese momento no es posible acceder a esa carpeta, como se puede ver en la *Figura 59*.

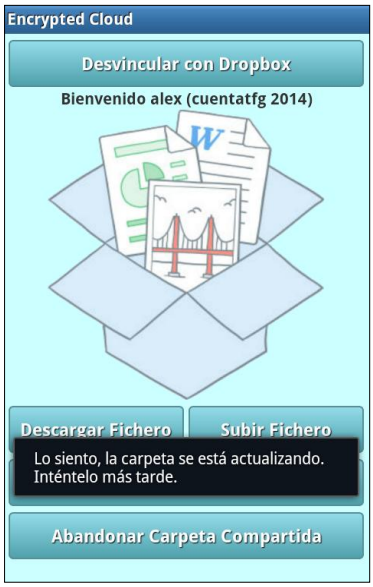


Figura 59. Error al subir fichero a carpeta compartida actualizándose

4.7.6.7. Descargar un fichero

Lo primero que se deberá hacer para poder descargar un fichero de una cuenta de *Dropbox* es pulsar sobre el botón "Descargar Fichero" que se encuentra en la actividad ***DropboxManager***, resaltado en la *Figura 60*.

Figura 60. Botón Descargar Fichero en *DropboxManager*

A continuación, se desplegará un explorador de carpetas, en el cual se van a listar todas las carpetas que hay disponibles en la cuenta de *Dropbox* que esté vinculada.

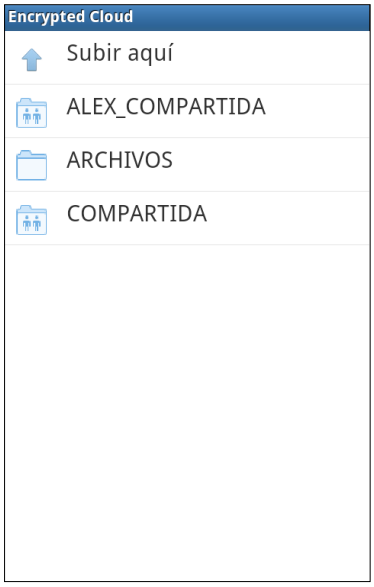


Figura 61. Explorador de carpetas *Dropbox* al descargar fichero

Este explorador mostrará todas las carpetas ya sean compartidas o no, pudiendo entrar en cada carpeta para poder observar su contenido, como se puede observar en la *Figura 62* en el caso de la carpeta "COMPARTIDA".

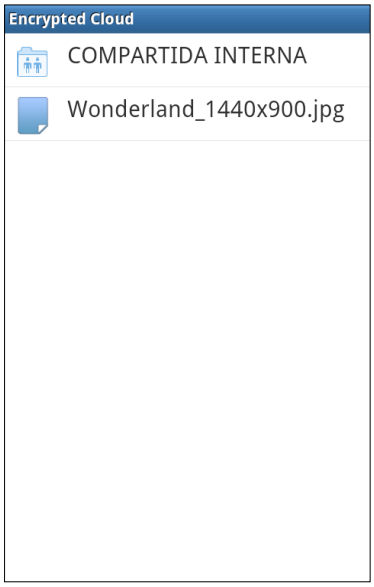


Figura 62. Explorador de carpetas interno *Dropbox* al descargar fichero

Una vez seleccionado el archivo que se quiere descargar, se procederá a su preparación para poder efectuar la descarga, como se muestra en la *Figura 63*.

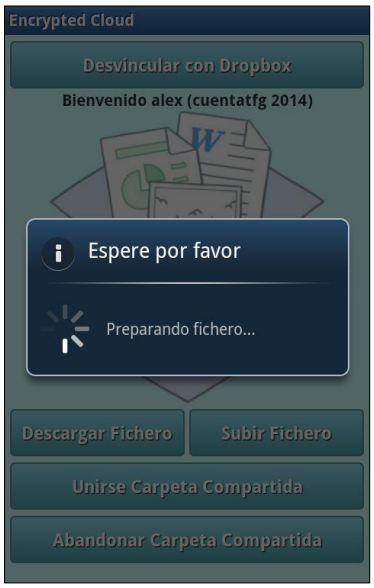


Figura 63. Preparación para descargar un fichero de *Dropbox*

Esta preparación del fichero va a consistir en que si éste se encuentra contenido en una carpeta compartida, sólo se procederá a su descarga en el caso de que el usuario actual se encuentre unido a dicha carpeta. En caso de que se encuentre en una carpeta no compartida, se realizará la descarga sin tener que hacer ninguna comprobación.

Cuando se ha terminado la preparación del archivo, se descargará su contenido y se procesará de dos formas diferentes:

- Si el fichero se encontraba contenido en una carpeta no compartida, primero se comprobará que está firmado correctamente. En este caso, se descargará y se accederá a su contenido que estará cifrado. A continuación, se obtendrá la clave AES 256 correspondiente, a partir de la creación de la semilla formada por la concatenación del nombre de usuario *logueado* y el identificador de la cuenta de *Dropbox* vinculada. Una vez obtenida esta clave, se podrá realizar el descifrado correctamente.
- En cambio, si se ha seleccionado un fichero contenido en una carpeta compartida, lo primero que también habrá que a hacer es comprobar su firma digital, es decir, se deberá contrastar que el firmante de dicho fichero es alguno de los usuarios contenidos en la carpeta compartida en ese momento. A continuación, habrá que interpretar el paquete contenido en ese archivo, para poder obtener la clave simétrica con la que se ha cifrado el contenido del fichero original. Este proceso ha sido explicado también en la sección *4.6.4.3*. Una vez obtenida la clave simétrica, se podrá descifrar el fichero correctamente.

En todo momento se le mostrará al usuario el progreso de su descarga solicitada, como se puede observar en la *Figura 64*.

Figura 64. Progreso de la descarga de un fichero de *Dropbox*

Una vez finalizada la descarga del archivo, se dará la opción de poder abrirlo con la aplicación más apropiada, para poder observar el resultado del descifrado, como se puede ver en la *Figura 65* y en la *Figura 66*. Todo fichero descargado se guarda en el dispositivo móvil, en la ruta *mnt/sdcard/EncryptedCloud*.

Figura 65. Selección de aplicación para visualizar archivo descifrado

Figura 66. Visualización de archivo descifrado descargado de *Dropbox*

Hay que tener presente que si un usuario intenta descargar un fichero desde una carpeta compartida a la cual no pertenece, se le notificará impidiendo que realice esa descarga.

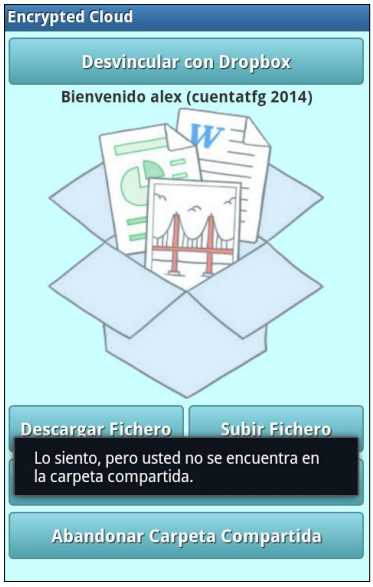


Figura 67. Error al descargar fichero de carpeta compartida

Además, si un usuario intenta descargar un archivo de una carpeta compartida cuando ésta se está actualizando, se le notificará que en ese momento no es posible acceder a esa carpeta, como se detalla en la *Figura 68*.

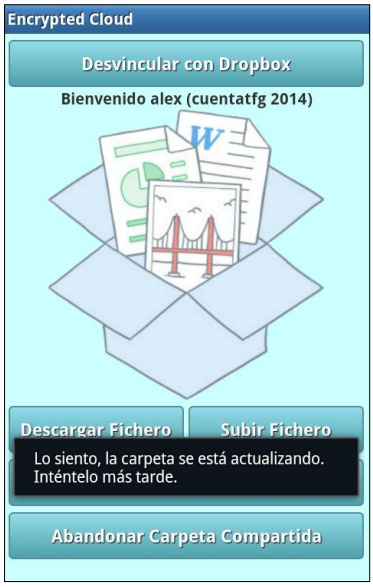


Figura 68. Error al descargar fichero a carpeta compartida actualizándose

Finalmente, como se ha comentado en párrafos anteriores, siempre que se descarga un archivo de una cuenta de *Dropbox*, lo primero que se comprueba es que ha sido firmado correctamente. En caso de que no lo fuera, la descarga se cancela y se le notifica al usuario del error encontrado (*Ver Figura 69*). Hay que tener en cuenta que el fichero sólo sería eliminado en el caso de que se encuentre en una carpeta compartida, y que el usuario actual sea el propietario de la misma.

Figura 69. Error al comprobar firma de fichero descargado de *Dropbox*

4.7.6.8. Control de ficheros subidos eliminados o movidos

Con el objetivo de poder controlar que los archivos subidos por un usuario a una cuenta de *Dropbox* no han sido movidos o eliminados por un tercero, cada vez que se realiza una subida, es registrada en la base de datos local de la aplicación, en la entidad ***FilesInPath***, tal y como se ha detallado en el apartado *4.7.6.6*. No obstante, hay que tener en cuenta que sólo se notifica que el fichero ha sido eliminado, sin posibilidad de

poder recuperar su contenido. Hay que destacar que sólo se monitorizan y registran aquellos ficheros subidos por el usuario a carpetas no compartidas y todos los ficheros encontrados en carpetas compartidas en las que es el propietario.

Cada vez que se inicia la actividad principal ***DropboxManager***, arranca un servicio encargado de comprobar si todos los registros que se encuentran en la entidad ***FilesInPath*** siguen estando en la ruta en la cual se subieron. En caso en el que no lo estén, se le notificará al usuario, tal y como se puede observar en la *Figura 70*.

Figura 70. Fichero eliminado indebidamente de una cuenta de *Dropbox*

4.8. Plan de pruebas

Una vez se terminó la implementación del proyecto, se comenzó con la fase de pruebas, con el fin de poder encontrar la máxima cantidad de errores para poder subsanarlos.

4.8.1. Tipos de pruebas

En cuanto a las pruebas realizadas, se pueden clasificar en dos grupos diferentes:

- **Pruebas unitarias**, utilizadas para poder probar el correcto funcionamiento de un módulo de código. Sirven para asegurar que cada uno de los módulos funcionan correctamente por separado [30].
- **Pruebas de integración**, que son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez [31].

4.8.2. Pruebas unitarias

Las pruebas unitarias realizadas se han encargado de comprobar el correcto funcionamiento de todas aquellas funcionalidades abarcadas por cada actividad individualmente, es decir, que no dependen de otra actividad para poder proporcionar dicha funcionalidad. Por ejemplo, la actividad ***Login*** es la encargada de realizar un *logueo* en la aplicación de un usuario, pero previamente se deberá haber creado una

cuenta de usuario, que es responsabilidad de la actividad ***Account***. Este tipo de prueba no sería unitaria, sino de integración.

Todas las pruebas unitarias han sido realizadas mediante la herramienta ***Robolectric***, la cual ha sido descrita en el apartado *4.6.2*.

4.8.2.1. Pruebas unitarias realizadas

A continuación se van a listar, por cada actividad de la aplicación implementada, las pruebas unitarias realizadas junto con una breve descripción:

- Las siguientes pruebas son comunes a las actividades ***Splash***, ***Login***, ***ModifyPassword***, ***Account*** y ***DropboxManager***.
    - ***TestActivityNotNull***. Comprueba que la actividad creada no es nula.
    - ***TestElementsNotNull***. Comprueba que los elementos de la interfaz de la actividad no son nulos.
    - ***TestTextElements***. Comprueba que los elementos de la interfaz tienen asociado el texto correcto.
- ***Splash***
    - ***TestClickLogin***. Comprueba que se carga la actividad ***Login*** cuando se pulsa sobre el botón "*Login*".
- ***Login***
    - ***TestClickCancel***. Comprueba que no se crea una nueva actividad cuando se pulsa sobre el botón "Cancelar".
    - ***TestClickModifyPassword***. Comprueba que se carga la actividad ***ModifyPassword*** cuando se pulsa sobre el botón "Modificar Contraseña".
    - ***TestClickNewUser***. Comprueba que se carga la actividad ***Account*** cuando se pulsa sobre el botón "Nuevo usuario".
- ***ModifyPassword***
    - ***TestClickCancel***. Comprueba que no se crea una nueva actividad cuando se pulsa sobre el botón "Cancelar".
    - ***TestModifyPasswordNewPassNotMatch***. Comprueba que se obtiene un error al modificar la contraseña de una cuenta de usuario, debido a que al repetir dos veces la nueva contraseña no coinciden.
    - ***TestModifyPasswordNewPassEqualsOldPass***. Comprueba que se obtiene un error al modificar la contraseña de una cuenta de usuario, debido a que la nueva contraseña es la misma que la antigua.
    - ***TestNewAccountNotEmptyFields***. Comprueba todos los posibles errores al modificar la contraseña de una cuenta de usuario, debido a que alguno de los campos a rellenar se encuentra vacío.
- ***Account***
    - ***TestClickCancel***. Comprueba que no se crea una nueva actividad cuando se pulsa sobre el botón "Cancelar".
    - ***TestNewAccountOK***. Comprueba que se puede crear una nueva cuenta de usuario.

- ***TestNewAccountUserRepeat***. Comprueba que no se puede crear una nueva cuenta si se proporciona como nombre de usuario uno ya existente en la base de datos.
- ***TestNewAccountPassNotMatch***. Comprueba que no se puede crear una nueva cuenta de usuario cuando al introducir la contraseña dos veces, éstas no coinciden.
- ***TestNewAccountNotEmptyFields***. Comprueba todos los posibles errores al crear una cuenta de usuario, debido a que alguno de los campos a rellenar se encuentra vacío.

4.8.2.2. Resultados de las pruebas unitarias realizadas

En total se realizaron 28 pruebas unitarias. Al principio, cómo es lógico, se obtuvieron varios errores que hubo que corregir. No obstante, se consiguió superar todas las pruebas diseñadas, como se puede observar en los resultados mostrados en el *Anexo D. Resultados de las pruebas unitarias realizadas.*

4.8.3. Pruebas de integración

En las pruebas de integración, se van a probar todas aquellas funcionalidades que proporciona la aplicación, y que no han sido verificadas previamente mediante las pruebas unitarias. A continuación se van a detallar las que son consideradas más importantes, adjuntando los resultados de cada una de ellas. No obstante, en el *Anexo E. Resultados de pruebas de integración realizadas*, se pueden encontrar otra serie de pruebas llevadas a cabo. Para realizar algunas de ellas, ha sido muy útil la herramienta ***LogCat*** contenida en el entorno de desarrollo.

LogCat es un sistema de *log* encargado de recolectar y ver las salidas de los *logs* generados por una aplicación, que son visualizados como mensajes, pudiendo realizar su filtrado [32].

Los mensajes de *log* en *Android* se van a clasificar por su criticidad, existiendo así varias categorías (ordenadas de mayor a menor criticidad) [33]:

1. *Error*.
2. *Warning*.
3. *Info*.
4. *Debug*.
5. *Verbose*.

Los mensajes usados en las pruebas de integración tienen una criticidad de *Info*.

4.8.3.1. Generación del par de claves RSA

Como se ha mencionado en apartados anteriores, el par de claves RSA asociado a un usuario y una cuenta de *Dropbox* es imprescindible para llevar a cabo los distintos cifrados. En la generación del par de claves hay dos enfoques, que son las dos pruebas que se han realizado.

Primero, cuando se realiza una nueva vinculación entre una cuenta de *Dropbox* y un usuario de la aplicación, se genera un nuevo par de claves RSA, almacenándolo en la base de datos en la tabla ***Accounts***. Para probar esta funcionalidad, lo que primero que se ha hecho es realizar una nueva vinculación a una cuenta de *Dropbox*. Justo en el momento en el que se le dan permisos a la aplicación "*EncryptedCloud* API", es cuando se realiza la generación del par de claves. En ese instante se comprobará que a través del ***LogCat*** aparece el mensaje "Nuevo par de claves RSA creado", como se puede observar en la *Figura 71*.

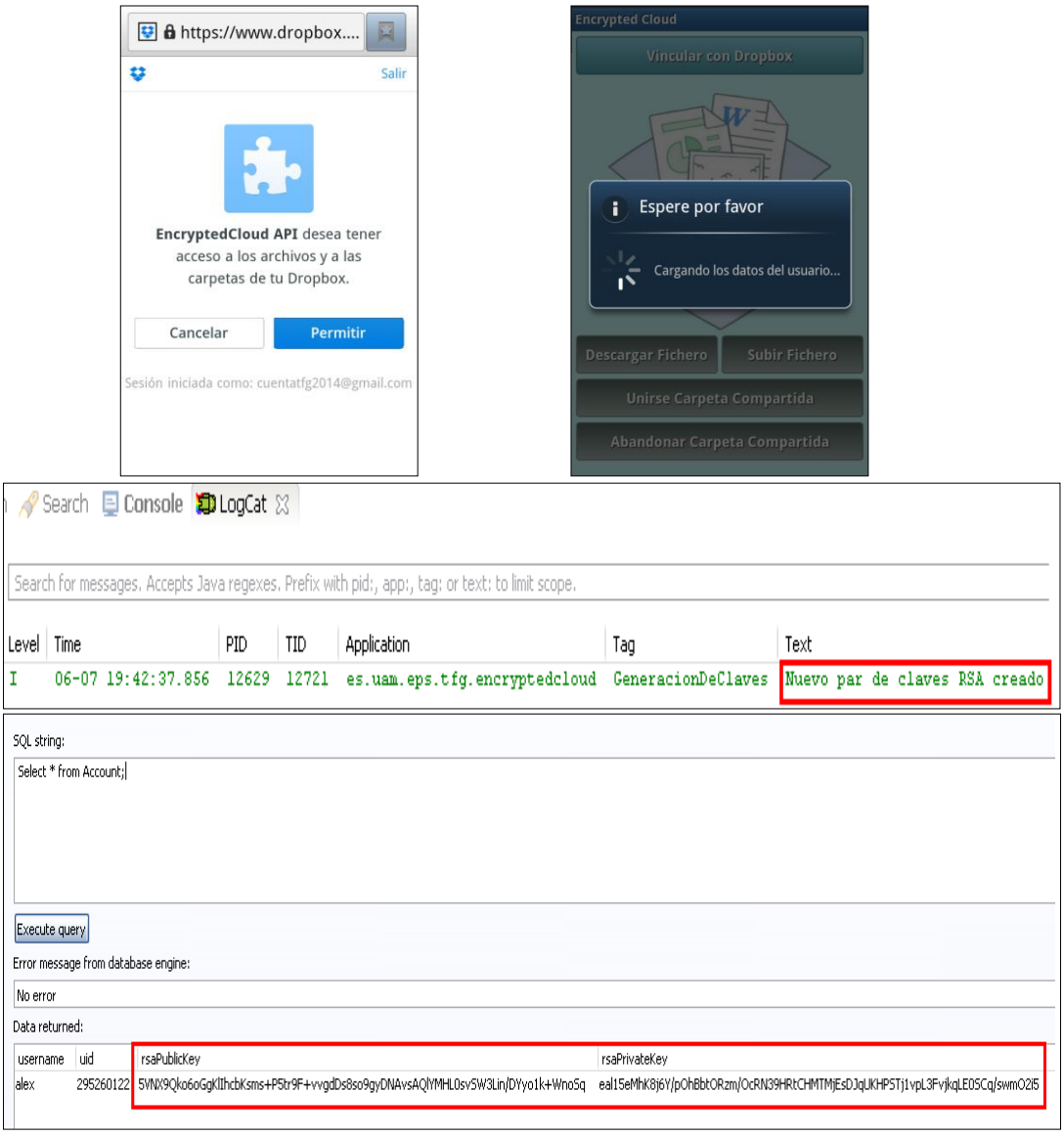


Figura 71. Resultados prueba generación de claves RSA (I)

Por otro lado, cuando se vuelve a vincular una cuenta de *Dropbox* a un usuario de la aplicación, únicamente se accede a la base de datos, más concretamente a la tabla ***Accounts***, para poder recuperar el par de claves RSA que fue generado y almacenado la primera vez que se hizo la vinculación. Para realizar esta prueba, primero se vincula un usuario con una cuenta de *Dropbox* a la que ya se haya vinculado anteriormente. En el momento de la vinculación en el cual se le dan permisos a la aplicación "*EncryptedCloud* API", es cuando se va a recuperar el par de claves de la base de datos.

En ese instante se comprobará a través del ***LogCat*** que aparece el mensaje "Par de claves RSA recuperado de la base de datos", como se puede ver en la *Figura 72*.

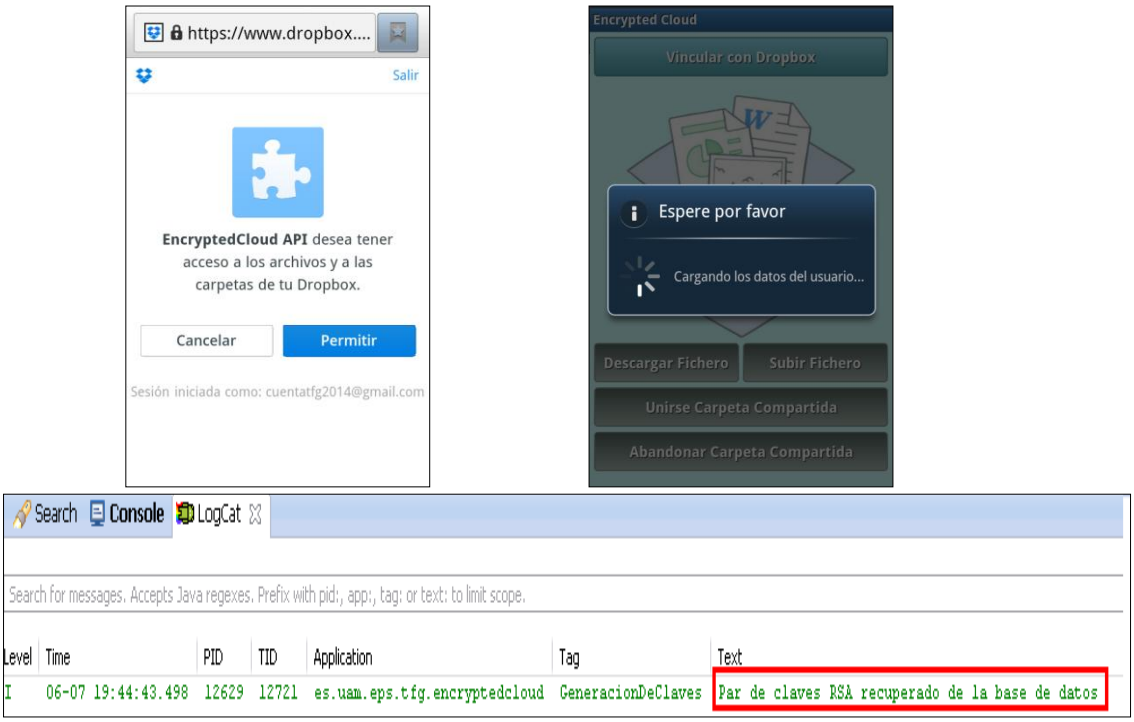


Figura 72. Resultados prueba generación de claves RSA (II)

4.8.3.2. Subida de un fichero a una cuenta de Dropbox

En esta sección se van a documentar diferentes pruebas que se han realizado al subir un fichero a una cuenta de *Dropbox*.

Para subir cualquier fichero, siempre se deberán seguir los mismos pasos. Estando en la actividad ***DropboxManager***, se va a pulsar sobre el botón "Subir Fichero". A continuación, se abrirá un explorador de archivos del dispositivo móvil, donde se tendrá que seleccionar el archivo que se desea subir (*Ver Figura 73*). Por último, se deberá seleccionar la carpeta de *Dropbox* donde se desea situar el fichero.

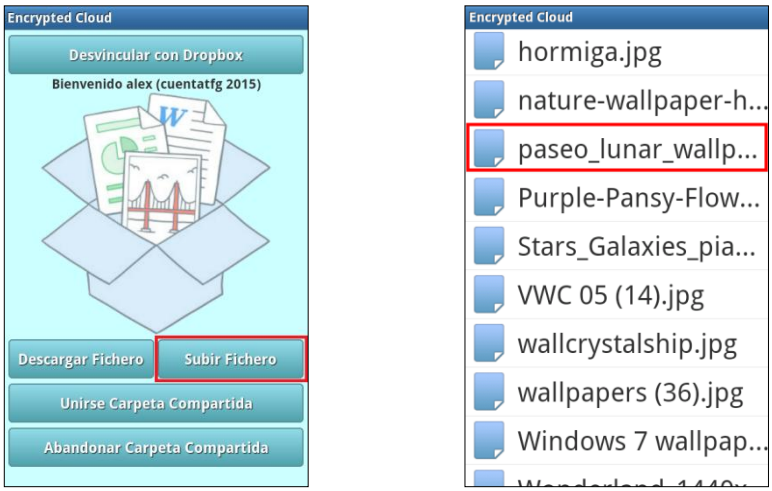


Figura 73. Resultados prueba subida fichero a *Dropbox* (I)

La primera prueba consistió en subir un fichero del dispositivo móvil a una carpeta no compartida de la cuenta de *Dropbox* vinculada. En este caso descrito en la *Figura 74*, se seleccionará la carpeta "ARCHIVOS" y posteriormente se verificará que el fichero se ha subido correctamente.

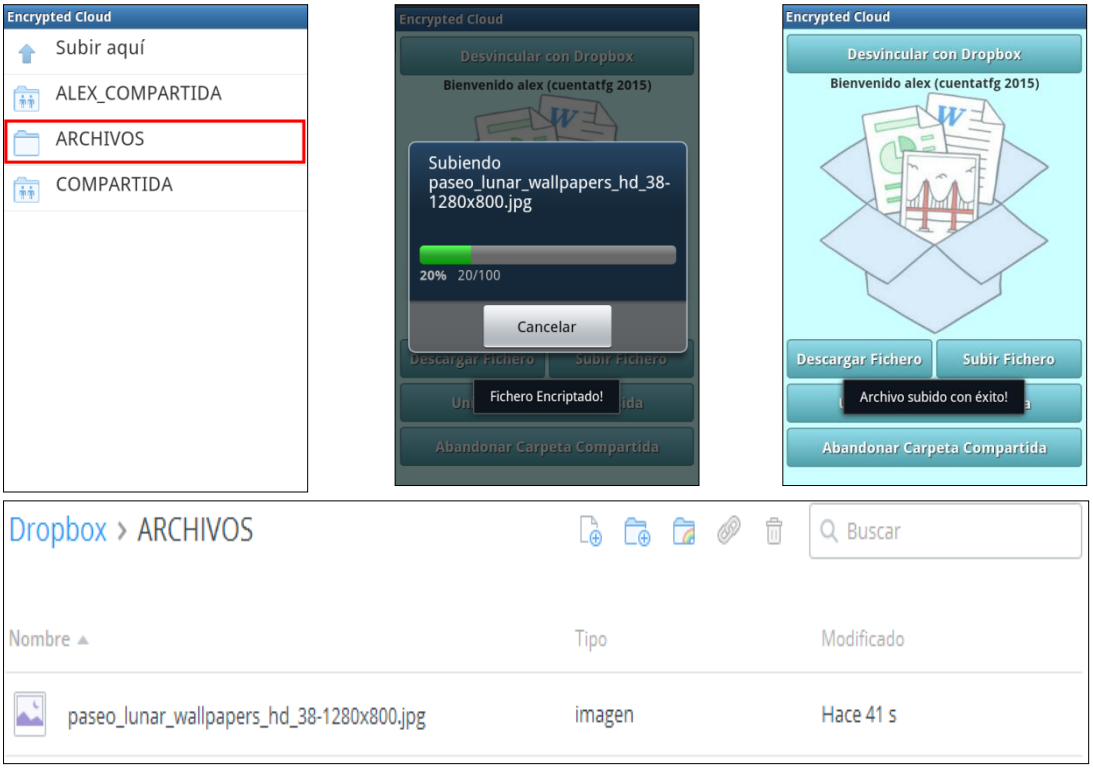


Figura 74. Resultados prueba subida fichero a *Dropbox* (II)

La segunda prueba tuvo el objetivo de subir un fichero a una carpeta compartida en la cual el usuario si estuviera unido. Para ello, estando previamente unido a esa carpeta, se seleccionó como directorio de subida la carpeta llamada "COMPARTIDA". Por último, se verificó que el archivo había sido subido correctamente, como se puede ver en la *Figura 75* y en la *Figura 76*.

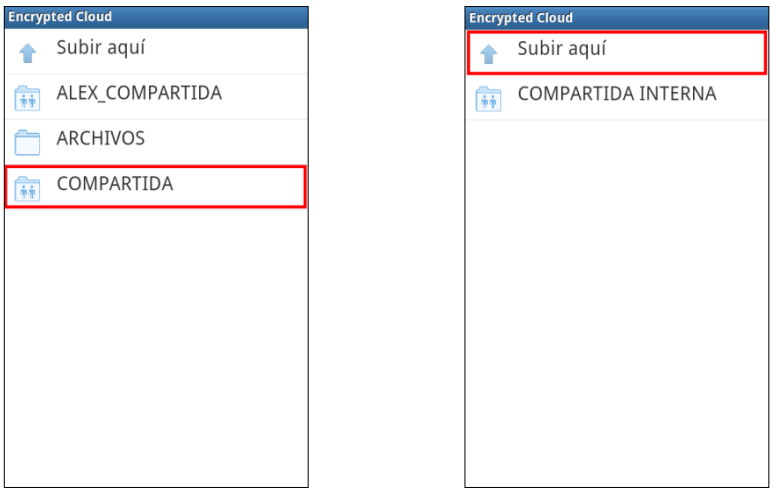


Figura 75. Resultados prueba subida fichero a *Dropbox* (III)

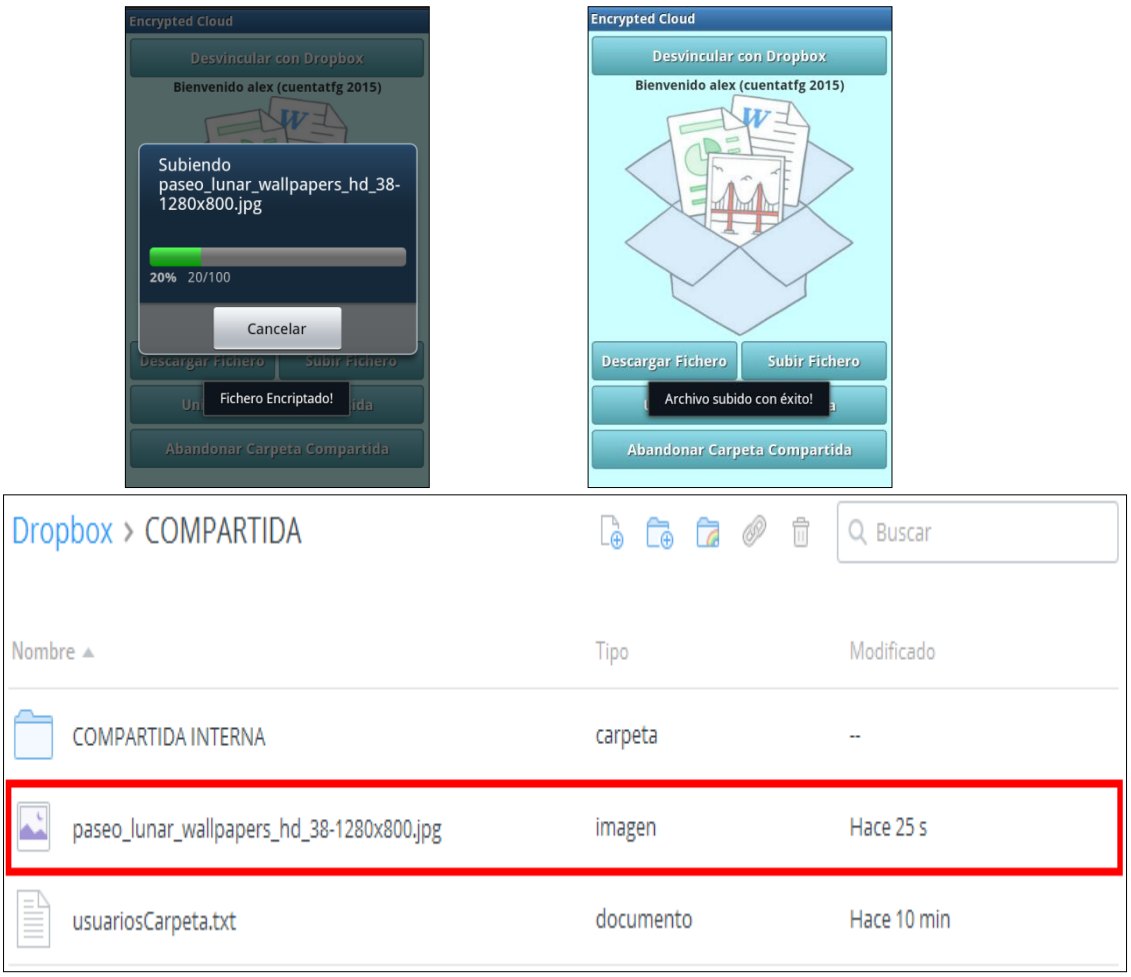


Figura 76. Resultados prueba subida fichero a *Dropbox* (IV)

Con la siguiente prueba se quiso comprobar que un usuario no puede subir un fichero a una carpeta compartida a la que no pertenece. Se seleccionó como carpeta destino la llamada "ALEX COMPARTIDA", a la cual el usuario actual no estaba unido. Se verificó que se le muestra un mensaje indicando que no se encuentra en la carpeta compartida, como el que aparece en la *Figura 77*.

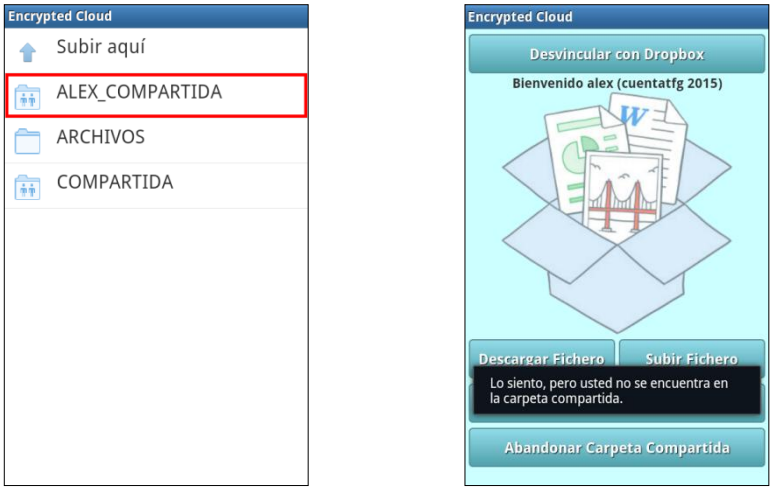


Figura 77. Resultados prueba subida fichero a *Dropbox* (V)

Por último, se comprobó que cuando una carpeta compartida está siendo actualizada por el propietario de la misma, no se le deja a ningún usuario poder subir un fichero a esa carpeta hasta que no termine esa actualización. En esta prueba, el usuario intenta realizar los mismos pasos que en la prueba dos, solo que al estar la carpeta actualizándose (esto es indicado mediante la subida a la carpeta compartida por parte del propietario de un fichero llamado *actualizandoCarpeta.txt*), se le va a impedir al usuario el acceso a dicha carpeta, mostrándose esta prueba en la *Figura 78*.

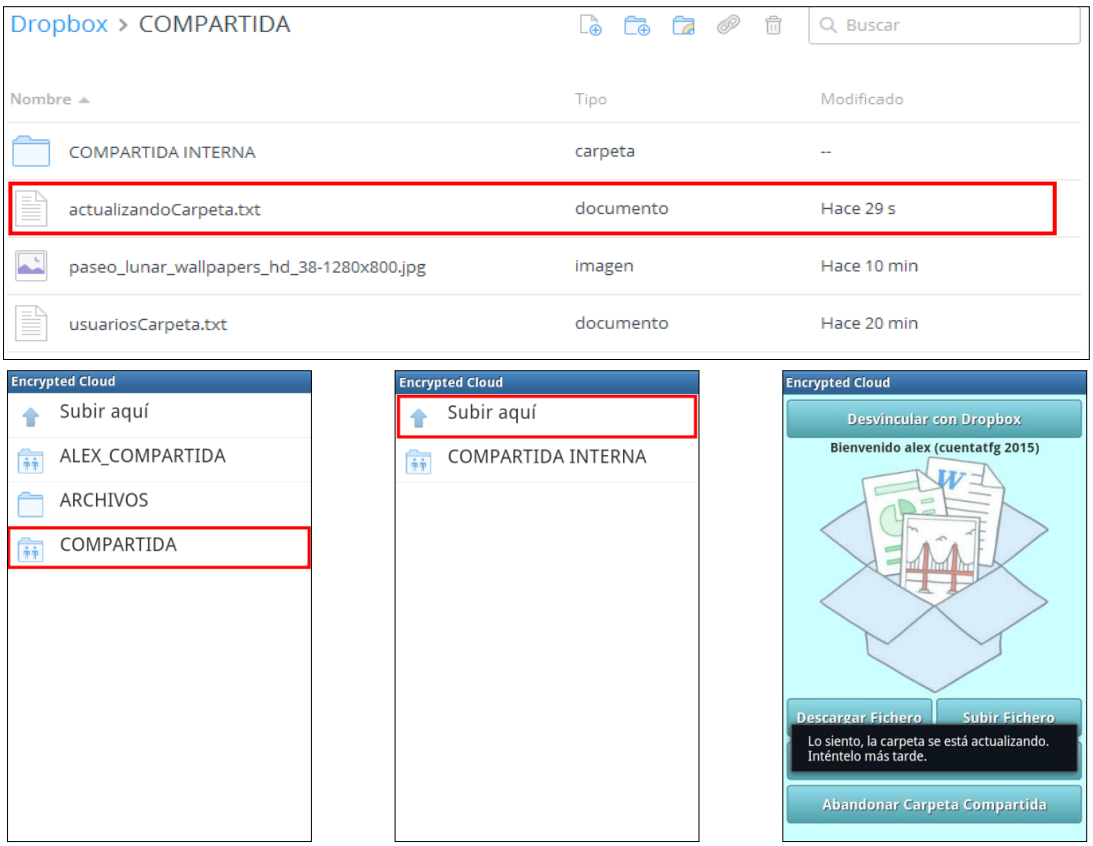


Figura 78. Resultados prueba subida fichero a *Dropbox* (VI)

4.8.3.3. Descarga de un fichero de una cuenta de Dropbox

En esta sección se van a documentar diferentes pruebas que se han realizado al descargar un fichero de una cuenta de *Dropbox*.

Para descargar cualquier fichero, siempre se deberán seguir los mismos pasos. Estando en la actividad ***DropboxManager***, se va a pulsar sobre el botón "Descargar Fichero" (*Ver Figura 79*). A continuación, se abrirá un explorador mostrando los archivos y carpetas contenidos en una cuenta de *Dropbox* vinculada. Se deberá ir navegando, hasta seleccionar el archivo que se desee descargar.

Figura 79. Resultados prueba descarga fichero de *Dropbox* (I)

La primera prueba consistió en descargar un fichero de a una carpeta no compartida de la cuenta de *Dropbox* vinculada. En este caso, se seleccionará la carpeta "ARCHIVOS" y posteriormente se elegirá el fichero "*paseo_lunar_wallpapers_h…*". En la *Figura 80* se puede comprobar que el fichero ha sido descargado correctamente, pudiéndose visualizar.

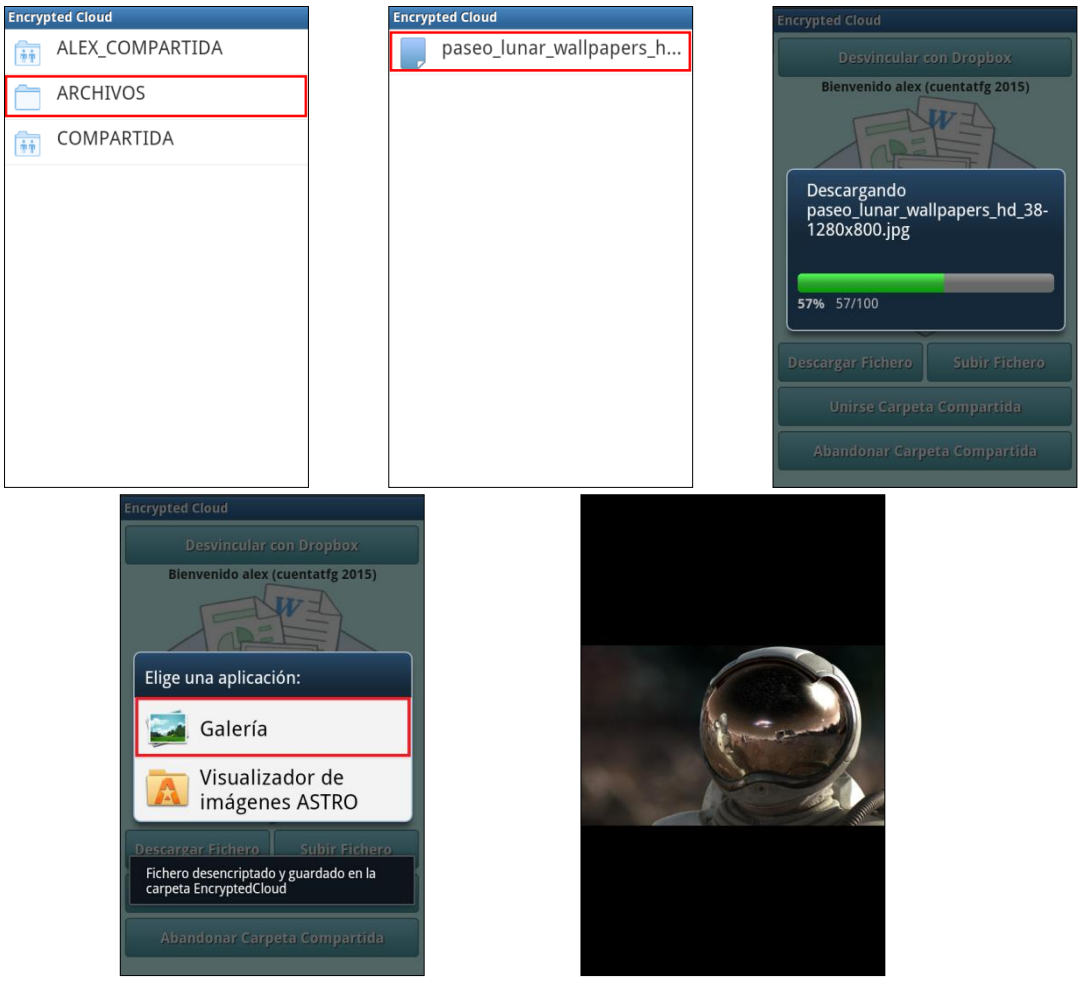


Figura 80. Resultados prueba descarga fichero de *Dropbox* (II)

La segunda prueba tuvo el objetivo de descargar un fichero de una carpeta compartida en la cual el usuario sí estuviera unido. Para ello, estando previamente unido a esa carpeta, se navegó hacia el contenido de la carpeta "COMPARTIDA". A continuación, se seleccionó el fichero "*paseo_lunar_wallpapers_h…*". En la *Figura 81* se puede comprobar que el fichero ha sido descargado correctamente, pudiéndose visualizar.

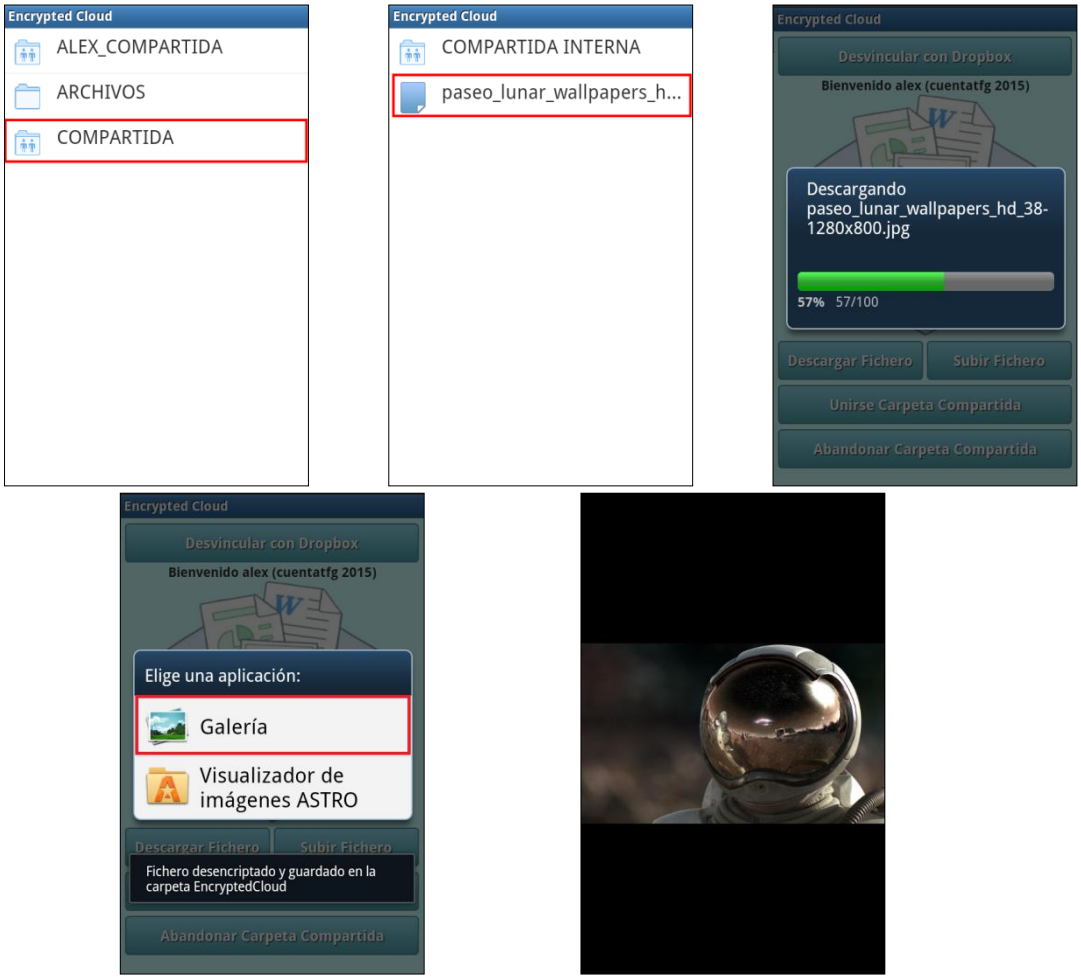


Figura 81. Resultados prueba descarga fichero de *Dropbox* (III)

Con la siguiente prueba se quiso comprobar que un usuario no puede descargar un fichero de una carpeta compartida a la que no pertenece. Se seleccionó el fichero "*paseo_lunar_wallpapers_h…*", contenido en la carpeta llamada "ALEX COMPARTIDA", a la cual el usuario actual no estaba unido. Se verificó que se le muestra un mensaje indicando que no se encuentra en la carpeta compartida, tal y como se puede ver en la *Figura 82*.

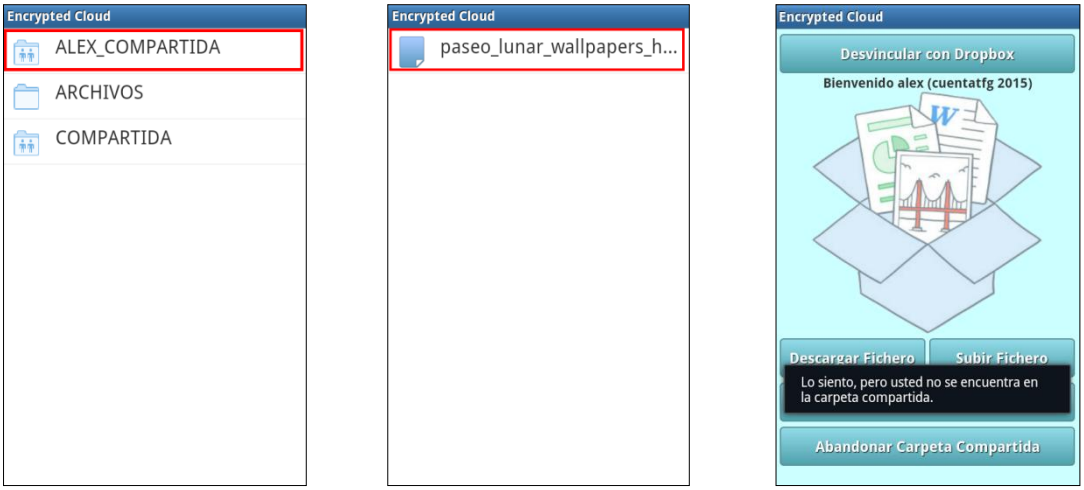


Figura 82. Resultados prueba descarga fichero de *Dropbox* (IV)

En la cuarta prueba, se comprobó que cuando una carpeta compartida está siendo actualizada por el propietario de la misma, no se le deja a ningún usuario poder descargar un fichero de esa carpeta hasta que no termine esa actualización. En esta prueba, el usuario intenta realizar los mismos pasos que en la prueba dos, solo que al estar la carpeta actualizándose, se le va a impedir al usuario el acceso a dicha carpeta.

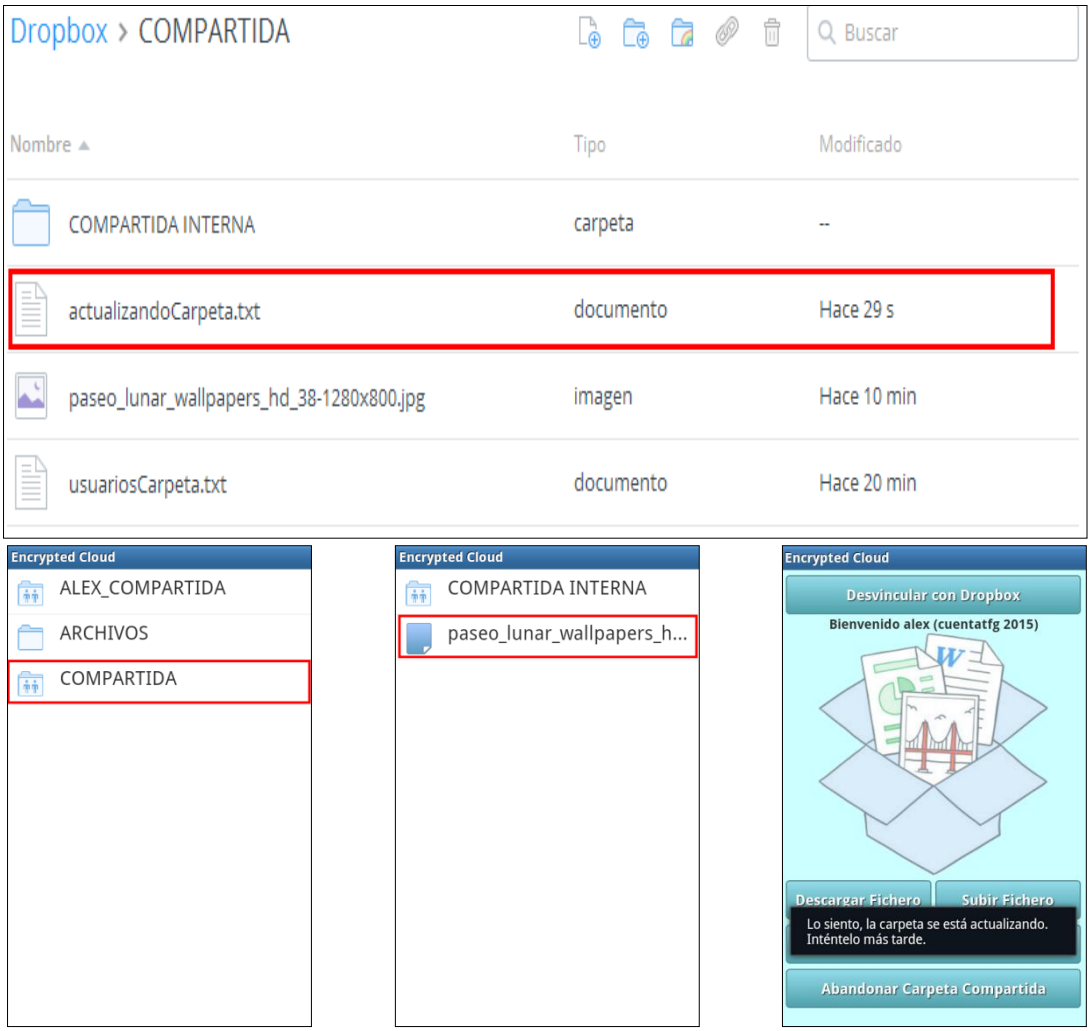


Figura 83. Resultados prueba descarga fichero de *Dropbox* (V)

Por último, se quiso comprobar que cuando se descarga un fichero sin firmar, se le notifica al usuario que la firma del fichero descargado no es correcta. Para poder probar este caso, primero se subió un fichero cualquiera a la cuenta de *Dropbox*. A continuación, se intenta realizar la descarga, comprobando que se le notifica al usuario que se ha encontrado un error en la firma de dicho fichero (*Ver Figura 84*). Además, hay que tener en cuenta que el fichero sería eliminado si se encontrase en una carpeta compartida, y el usuario fuese el propietario de la misma.

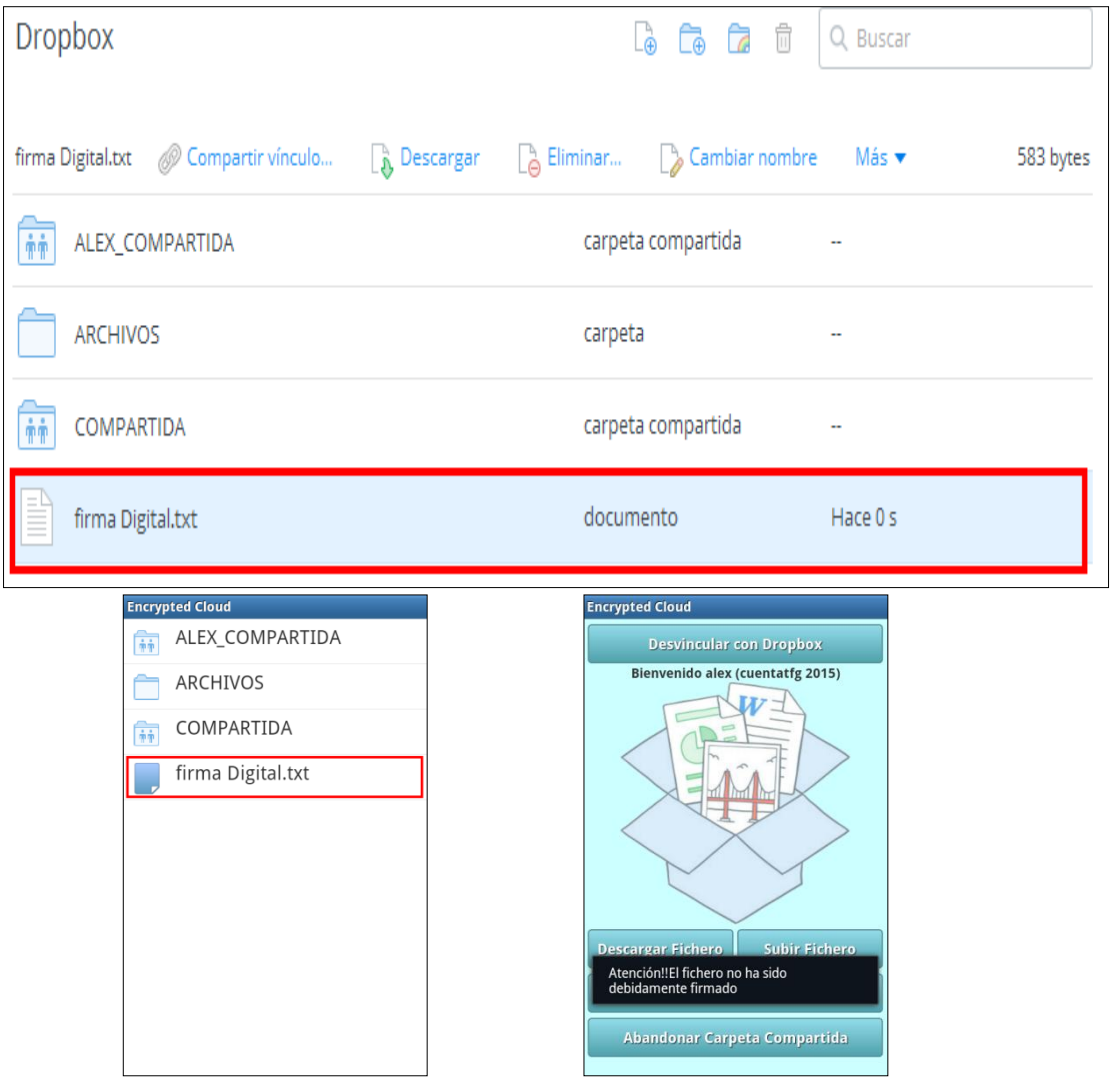


Figura 84. Resultados prueba descarga fichero de *Dropbox* (VI)

Se puede comprobar, observando la *Figura 85*, que todas las descargas realizadas se guardan en la ruta */mnt/sdcard/EncryptedCloud*.

Figura 85. Resultados prueba descarga fichero de *Dropbox* (VII)

4.8.3.4. Unirse a una carpeta compartida

En este apartado se van a detallar las pruebas realizadas relacionadas con la unión de un usuario a una carpeta compartida. Para realizar esta acción, se tendrá que pulsar sobre el botón "Unirse Carpeta Compartida", contenido en la actividad ***DropboxManager*** (*Ver Figura 86*). A continuación, se abrirá un explorador mostrando las carpetas compartidas contenidas en una cuenta de *Dropbox* vinculada.

Figura 86. Resultados prueba unirse carpeta compartida (I)

Para empezar, se comprobó que cuando una carpeta compartida se está actualizando, no es posible unirse a ella. Para llevar a cabo esta prueba, se seleccionó la carpeta "COMPARTIDA", la cual se estaba actualizando en ese momento, tal y como se muestra en la *Figura 87*.

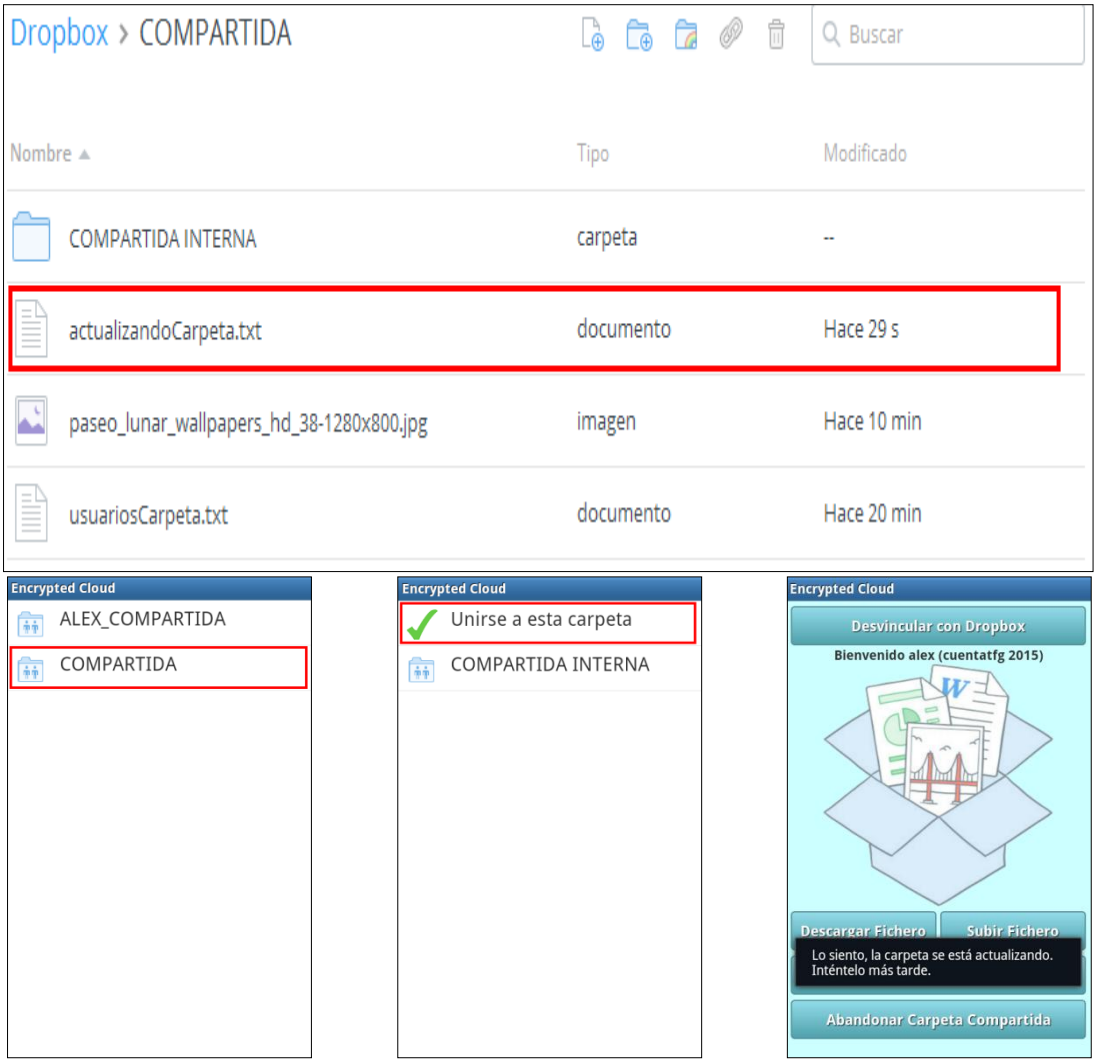


Figura 87. Resultados prueba unirse carpeta compartida (II)

Las siguientes pruebas se realizaron a partir de dos enfoques distintos desde el punto de vista del usuario, ya que puede ser o no propietario de una carpeta compartida, dependiendo de si al unirse a una carpeta hay ya usuarios o no.

Primero, se probó la unión de un propietario a una carpeta compartida. Para ello, se seleccionó la carpeta "ALEX_COMPARTIDA", comprobando que se le muestra al usuario un mensaje indicándole que se ha unido sin problemas. Esta prueba se puede ver en la *Figura 88*.

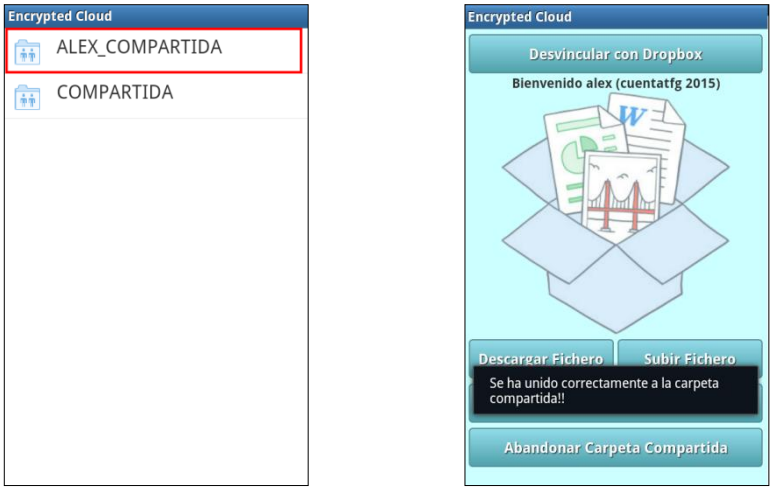


Figura 88. Resultados prueba unirse carpeta compartida (III)

A continuación, se repitió el procedimiento realizado en la anterior prueba, con el fin de comprobar que un usuario no puede unirse dos veces a una carpeta compartida. En la *Figura 89* se verificó que se le muestra al usuario un mensaje indicándole que ya se encuentra en la carpeta compartida.

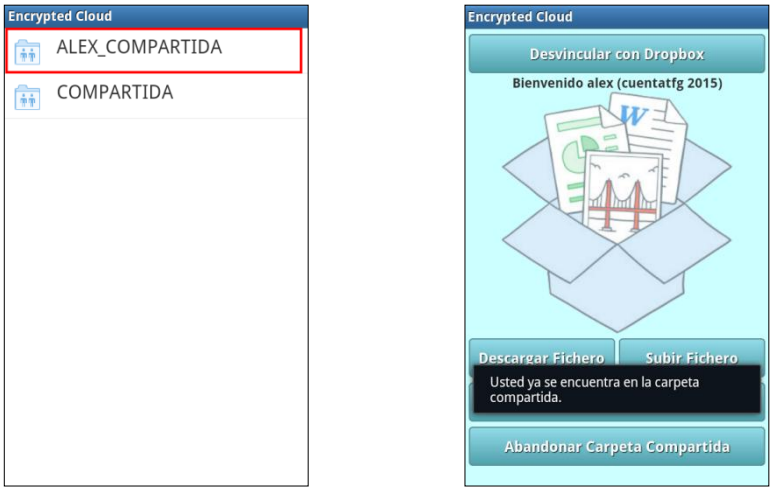


Figura 89. Resultados prueba unirse carpeta compartida (IV)

Desde el punto de vista de un usuario que no es propietario, se quiso comprobar que cuando se quiere unir a una carpeta compartida, va a mandar una petición a su propietario, subiéndola a una carpeta llamada "*peticionesUnirseCarpeta*" contenida en la carpeta compartida a la que se quiere unir. Para ello, se seleccionó la carpeta "ALEX_COMPARTIDA", a la cual se mandó una petición de unión, como se puede ver en la *Figura 90*.

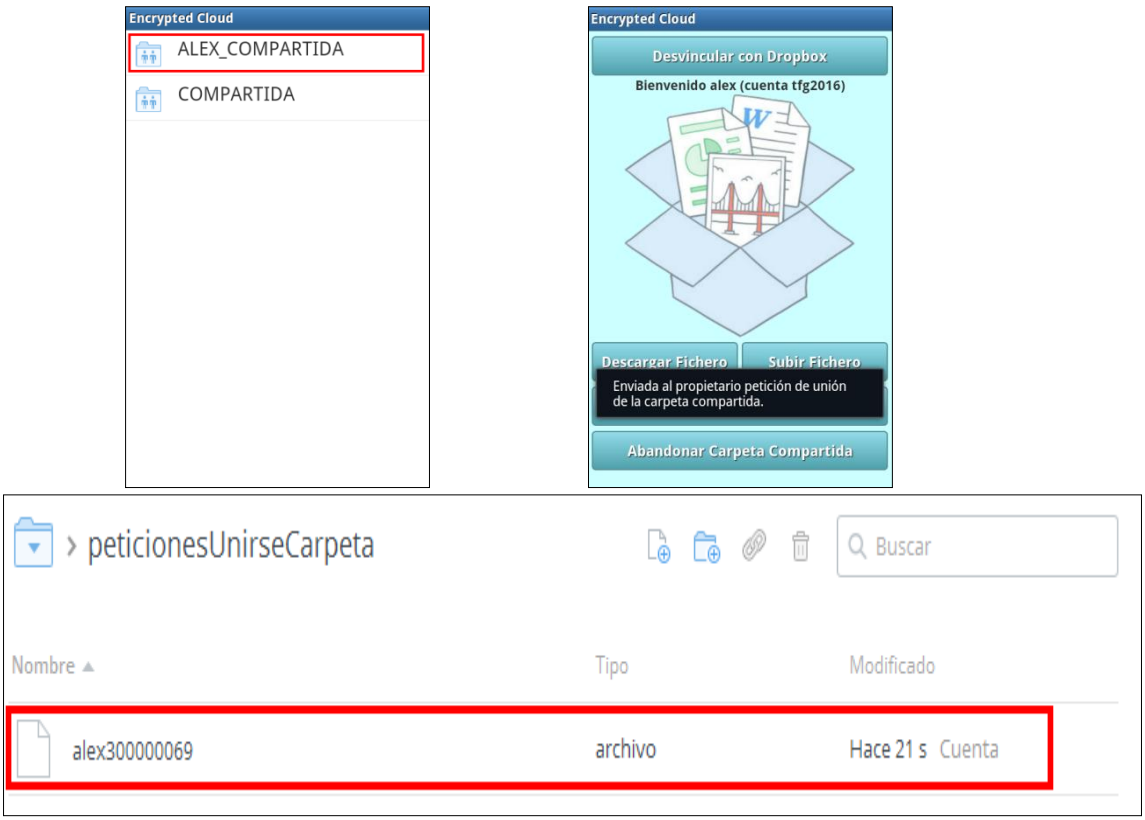


Figura 90. Resultados prueba unirse carpeta compartida (V)

Además, se quiso comprobar que si se elimina dicha petición, se vuelve a subir sin ningún tipo de problema, pudiéndose verificar en la *Figura 91*.

Figura 91. Resultados prueba unirse carpeta compartida (VI)

4.8.3.5. Abandonar una carpeta compartida

En esta sección se van a detallar las pruebas realizadas relacionadas con el abandono de un usuario de una carpeta compartida. Para realizar esta acción, se tendrá que pulsar sobre el botón "Abandonar Carpeta Compartida", contenido en la actividad ***DropboxManager*** y resaltado en la *Figura 92*. A continuación, se abrirá un explorador mostrando las carpetas compartidas contenidas en una cuenta de *Dropbox* vinculada.

Figura 92. Resultados prueba abandonar carpeta compartida (I)

Para empezar, se comprobó que cuando una carpeta compartida se está actualizando, no es posible abandonarla. Para llevar a cabo esta prueba, se seleccionó la carpeta "COMPARTIDA", la cual se estaba actualizando en ese momento. Este proceso es ilustrado en la *Figura 93* y en la *Figura 94*.

Figura 93. Resultados prueba abandonar carpeta compartida (II)

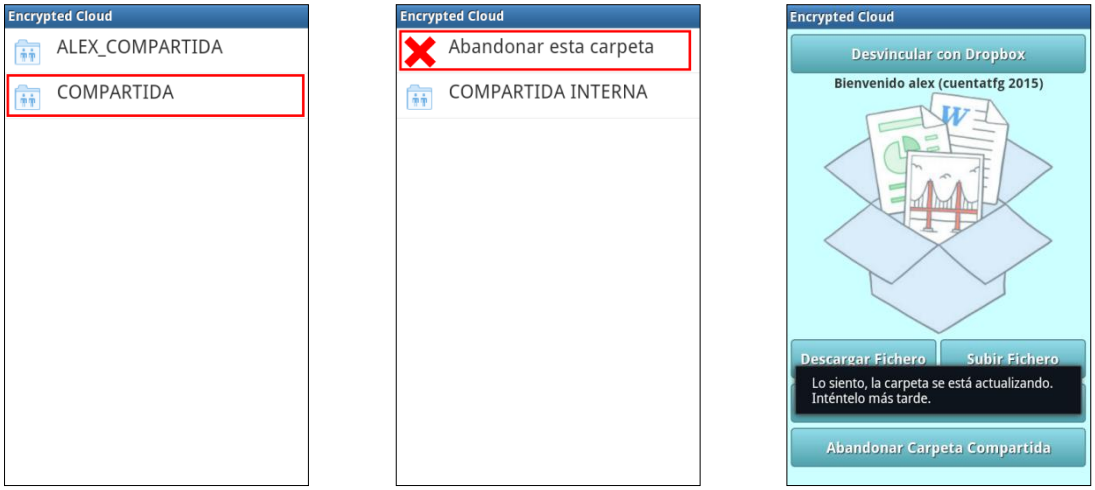


Figura 94. Resultados prueba abandonar carpeta compartida (III)

Las siguientes pruebas se realizaron a partir de dos enfoques distintos desde el punto de vista del usuario, ya que puede ser o no propietario de una carpeta, dependiendo de si al unirse a una carpeta hay ya usuarios o no.

Primero, se probó el abandono de un propietario de una carpeta compartida. Para ello, se seleccionó la carpeta "ALEX_COMPARTIDA", comprobando que se le muestra al usuario un mensaje indicándole que ha abandonado la carpeta sin problemas, como se puede ver en la *Figura 95* y en la *Figura 96*. Además, también se comprueba que al abandonar el propietario la carpeta, se eliminan todos los archivos de la misma.

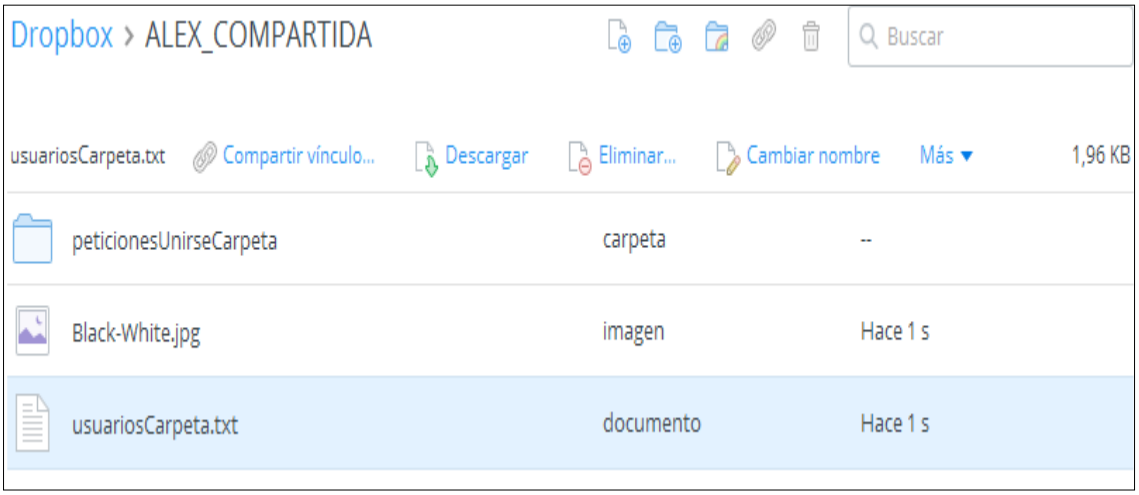


Figura 95. Resultados prueba abandonar carpeta compartida (IV)

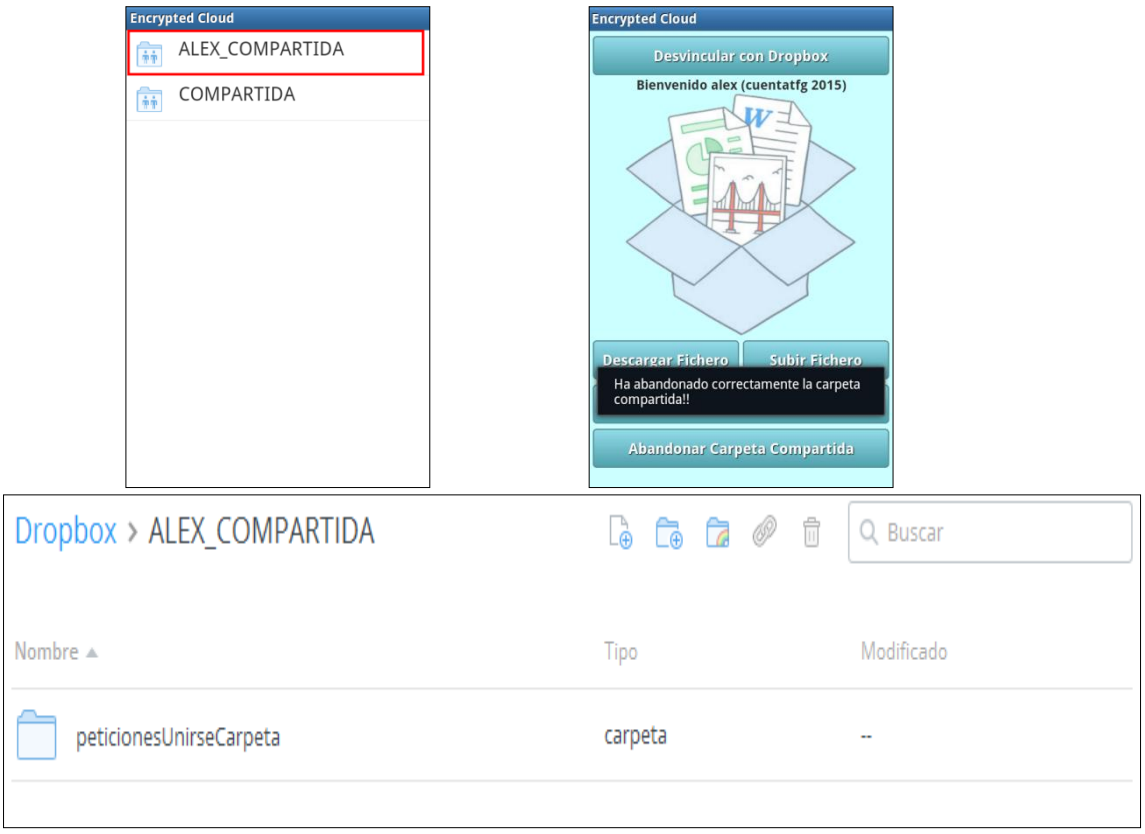


Figura 96. Resultados prueba abandonar carpeta compartida (V)

A continuación, se repitió el procedimiento realizado en la anterior prueba, con el fin de comprobar que un usuario no puede abandonar dos veces una carpeta compartida. En la *Figura 97* se verificó que se le muestra un mensaje indicándole que no se encuentra en la carpeta compartida.

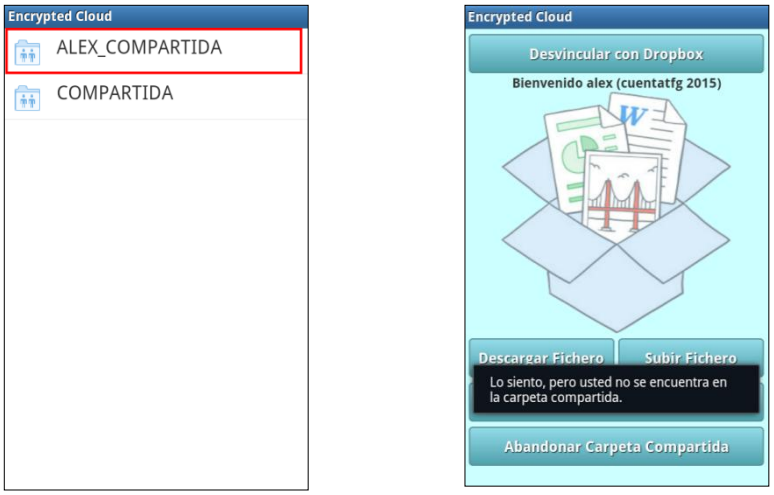


Figura 97. Resultados prueba abandonar carpeta compartida (VI)

Desde el punto de vista de un usuario que no es propietario, se quiso comprobar que cuando quiere abandonar una carpeta compartida, va a mandar una petición a su propietario, subiéndola a una carpeta llamada "*peticionesAbandonoCarpeta*" contenida en la carpeta compartida a la que se quiere unir. Para ello, se seleccionó la carpeta

"ALEX_COMPARTIDA", a la cual se mandó una petición de abandono. Este proceso se puede observar en la *Figura 98*.

Figura 98. Resultados prueba abandonar carpeta compartida (VII)

Además, se quiso comprobar que si se elimina dicha petición, se vuelve a subir sin ningún tipo de problema, tal y como se puede ver en la *Figura 99*.

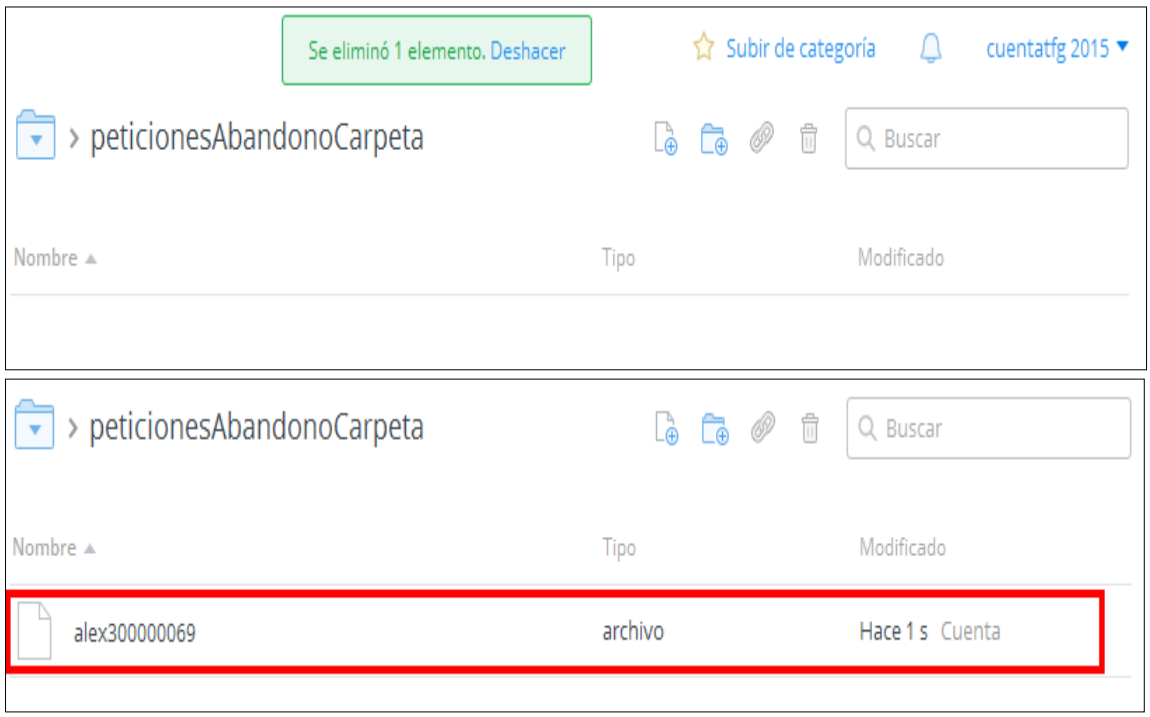


Figura 99. Resultados prueba abandonar carpeta compartida (VIII)

Capítulo 5
Conclusión y trabajo futuro

El objetivo principal del trabajo realizado era estudiar el problema existente en la distribución de claves criptográficas en entornos *cloud*. Esta necesidad surge debido a que cada vez son más los usuarios que llevan todos sus activos a la nube, y en algunas ocasiones desconfían sobre dónde se van a encontrar sus datos y quiénes van a tener acceso a ellos. Es por este motivo por el cual muchos usuarios sienten la obligación de tener que subir sus datos cifrados a la nube. El problema está cuando se quiere compartir un archivo cifrado con otro usuario, ya que se tendría que realizar una distribución de claves de manera segura, que es el fruto de la investigación de este trabajo.

Este estudio ha sido plasmado a lo largo de todo este documento. En primer lugar, se describió qué es un entorno *cloud* y cuáles son sus características principales, además de su estado en la actualidad. A continuación, se describieron varios aspectos considerados muy importantes y relacionados con la seguridad en la nube. Posteriormente, se presentó la solución implementada para poder solventar el problema de la distribución de claves en un entorno *cloud*, *Dropbox*. Para ello, se describió todas las bases en las que se apoya la aplicación *Android* desarrollada, haciendo hincapié en las relacionadas con la seguridad. También se detalló al completo toda la funcionalidad que ofrece y qué pasos hay que seguir para usarla. Por último, se especificó el plan de pruebas diseñado y realizado para verificar todas las características que debía cumplir la aplicación.

Tras esto, se puede concluir que el proyecto desarrollado cumple con los requisitos que inicialmente se plantearon, resolviendo satisfactoriamente el problema de distribución de claves criptográficas simétricas en *Dropbox*, un entorno *cloud* bastante popular.

Sin embargo, esta implementación tiene la posibilidad de ser mejorada en varios aspectos, que podrían ser implementados en un trabajo futuro. Para empezar, estaría bien implementar la aplicación en los sistemas operativos más utilizados de un dispositivo móvil y no sólo en *Android*. Además, se podría dar la opción de seleccionar un entorno *cloud* entre los más populares, ya que el protocolo de distribución de claves implementado es válido en cualquiera de ellos. También sería interesante el poder recuperar archivos que han sido subidos y posteriormente han sido modificados o eliminados. Por otro lado, hay que destacar que en ningún momento se ha evaluado la eficiencia del proyecto implementado. En un trabajo futuro, este aspecto se podría mejorar incorporando por ejemplo árboles de *Merkle* [34], ya que se reduciría el número de firmas digitales realizadas. Hay que decir que las operaciones realizadas con claves RSA de 4096 bits son muy costosas computacionalmente. Para mitigar este problema, se podría hacer uso de la criptografía de curvas elípticas [35], que reducen

considerablemente el tamaño de la clave proporcionando un nivel de seguridad equivalente.

Por último, desde el punto de vista más personal, hay que destacar que se han adquirido y afianzado muchos conocimientos sobre seguridad, siendo aprendidos a lo largo de estos cuatro cursos en diversas asignaturas. Además, se tiene una visión general sobre el problema de seguridad existente en la nube, el cual es un tema que cada vez va a dar más que hablar, debido a la gran tendencia que se está produciendo en los últimos años de subir gran parte de los activos a este entorno. En cuanto a *Android*, se han adquirido grandes destrezas de programación en esta plataforma, aspecto que es muy importante en estos tiempos, cuando la inmensa mayoría de los dispositivos móviles disponen de este sistema operativo.

Glosario

Activo. Recurso o bien económico propiedad de una entidad [36].

AES. Esquema de cifrado por bloques, que fue adoptado como estándar de cifrado por el gobierno estadounidense. Reemplaza progresivamente a su predecesor (DES y Triple DES), siendo uno de los algoritmos más utilizados en criptografía simétrica [37].

***Android Developer Tools* (ADT).** Plug-in de desarrollo en *Eclipse IDE* para el desarrollo de aplicaciones en *Android* [38].

API. Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción [39].

***Backup*.** Copia total o parcial de información del disco duro, bases de datos u otros medios de almacenamiento [40].

Firma Digital. Mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la entidad originadora de dicho mensaje y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador [25].

Firmware. Bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo [41].

Hash. Algoritmo que consigue crear a partir de una entrada una salida alfanumérica de longitud normalmente fija que representa un resumen de toda la información que se le ha dado [27].

Log. Registro oficial de eventos durante un rango de tiempo en particular [42].

Instituto Nacional de Normas y Tecnología (NIST). Agencia de la Administración de Tecnología del Departamento de Comercio de los Estados Unidos, cuya misión es promover la innovación y la competencia industrial en Estados Unidos mediante avances en metrología, normas y tecnología de forma que mejoren la estabilidad económica y la calidad de vida [43].

OAEP. Tipo de *padding* aplicado en conjunto con el cifrado de clave asimétrica RSA [44].

Open Source. *Software* distribuido y desarrollado libremente [45].

Padding. Esquema de relleno de mensajes usado en criptografía.

PKCS#5. Establece un sistema de rellenado (*padding*) para cifrados de bloque que consiste en rellenar con tantos bytes como se necesiten y el valor de los bytes será el mismo que el número de los bytes introducidos [46].

Plug-in. Aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al *software* [47].

RF. Requisito funcional.

RNF. Requisito no funcional.

RSA. Sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente [48].

***Smartphone*.** Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional [49].

Software. Equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados *hardware* [50].

TIC. Las tecnologías de la información y la comunicación (TIC) agrupan los elementos y las técnicas usadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, Internet y telecomunicaciones [51].

Bibliografía

[1] Ministerio de Economía y Competitividad de España, "Plan estatal de investigación científica, técnica y de innovación 2013-2016." [Online]. Available: http://www.idi.mineco.gob.es/stfls/MICINN/Investigacion/FICHEROS/Politicas_I+D+i/Plan_Estatal_Inves_cientifica_tecnica_innovacion.pdf. [Accessed: 19-Jun-2014].

[2] European Commission, "EU Cybersecurity plan to protect open internet and online freedom and opportunity - Cyber Security strategy and Proposal for a Directive," 2013. [Online]. Available: http://ec.europa.eu/digital-agenda/en/news/eu-cybersecurity-plan-protect-open-internet-and-online-freedom-and-opportunity-cyber-security. [Accessed: 15-Jun-2014].

[3] Cloudfogger, "Cloudfogger - Free File Encryption for Dropbox and the Cloud," 2012. [Online]. Available: http://www.cloudfogger.com/en/. [Accessed: 16-Jun-2014].

[4] SafeMonk, "SafeMonk | Dropbox + Encryption Made Easy." [Online]. Available: https://www.safemonk.com/. [Accessed: 16-Jun-2014].

[5] J. M. Fernández Granda, "Seguridad En La Nube," 2012. [Online]. Available: http://www.compartolid.es/wp-content/uploads/Seguridad-en-la-nube.pdf. [Accessed: 23-Jun-2014].

[6] O. A. Mejía, "Computación en la nube," *ContactoS*, vol. 80, pp. 45–52, 2011.

[7] P. Mell and T. Grance, "The NIST definition of cloud computing," *Spec. Publ. 800-145, NIST*, 2011.

[8] J. Collahuazo and J. Alexander, "Guía para el análisis de factibilidad en la implantación de tecnologías de Cloud Computing en empresas del Ecuador," QUITO/EPN/2012, 2012.

[9] Wikipedia, "Computación en la nube." [Online]. Available: http://es.wikipedia.org/wiki/Computación_en_la_nube#Comienzos. [Accessed: 05-Mar-2014].

[10] C. Boyd, "Cryptography in the Cloud: Advances and Challenges," *J. Inf. Commun. Converg. Eng.*, vol. 11, no. 1, pp. 17–23, Mar. 2013.

[11] R. Chandramouli, M. Iorga, and S. Chokhani, "Cryptographic Key Management Issues and Challenges in Cloud Services," in *Secure Cloud Computing*, S. Jajodia, K. Kantt, P. Samarati, P. Singhal, V. Swarup, and C.Wang, Eds. Springer, 2014, pp. 1–30.

[12] A. Juels and B. S. Kaliski Jr, “PORs: Proofs of retrievability for large files,” in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 584–597.

[13] K. D. Bowers, M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, “How to tell if your cloud files are vulnerable to drive crashes,” in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 501–514.

[14] Wikipedia, “Dropbox.” [Online]. Available: http://es.wikipedia.org/wiki/Dropbox. [Accessed: 02-Jun-2014].

[15] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and A. Ribagorda, “Evolution, Detection and Analysis of Malware for Smart Devices,” *Commun. Surv. Tutorials, IEEE*, vol. 16, no. 2, pp. 961–987, 2014.

[16] A. Vilchez, “Que es Android: Características y Aplicaciones.” [Online]. Available: http://www.configurarequipos.com/doc1107.html. [Accessed: 02-Jun-2014].

[17] RevistaSIC, “Ciberseguridad basada en la nube.” [Online]. Available: http://revistasic.es/index.php?option=com_content&view=article&id=1032&Itemid=919. [Accessed: 21-Jun-2014].

[18] Wikipedia, “Modelo entidad-relación.” [Online]. Available: http://es.wikipedia.org/wiki/Modelo_entidad-relación. [Accessed: 03-Jun-2014].

[19] Equipo docente de la asignatura Desarrollo de Aplicaciones en Dispositivos Móviles Universidad Autónoma de Madrid, “El Entorno de Desarrollo de Android y tu primera app,” pp. 1–18, 2013.

[20] J. F. Ávila de Tomás, “¿Qué es Mendeley?,” 2012. [Online]. Available: http://nuevastecsomamfyc.wordpress.com/2012/04/11/que-es-mendeley/. [Accessed: 03-Jun-2014].

[21] “android-string-resource-translator - String Resource Translator for Android - Google Project Hosting.” [Online]. Available: https://code.google.com/p/android-string-resource-translator/. [Accessed: 03-Jun-2014].

[22] R. Aguilera Díaz-Heredero, “Robolectric: aplicando TDD en Android,” 2012. [Online]. Available: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Robolectric. [Accessed: 06-Jun-2014].

[23] “Tutorial de JCA (Java Cryptographic Architecture).” [Online]. Available: http://ccia.ei.uvigo.es/docencia/SSI-grado/1213/practicas/tutorial-jca/index.html. [Accessed: 03-Jun-2014].

[24] “Spongy Castle.” [Online]. Available: http://rtyley.github.io/spongycastle/. [Accessed: 03-Jun-2014].

[25] Wikipedia, “Firma digital.” [Online]. Available: http://es.wikipedia.org/wiki/Firma_digital. [Accessed: 04-Jun-2014].

[26] “Firma Digital,” 2009. [Online]. Available: http://notiactual21.blogspot.com.es/2009/04/firma-digitalbusiness-intelligence.html. [Accessed: 04-Jun-2014].

[27] P. Gutiérrez, “¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales,” 2013. [Online]. Available: http://www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales. [Accessed: 04-Jun-2014].

[28] Wikipedia, “SHA-1.” [Online]. Available: http://en.wikipedia.org/wiki/SHA-1. [Accessed: 03-Jun-2014].

[29] L. Castro, “Qué es SSL - Guía básica sobre qué es SSL y qué significa.” [Online]. Available: http://aprenderinternet.about.com/od/ConceptosBasico/a/Que-Es-Ssl.htm. [Accessed: 04-Jun-2014].

[30] Wikipedia, “Prueba unitaria.” [Online]. Available: http://es.wikipedia.org/wiki/Prueba_unitaria. [Accessed: 05-Jun-2014].

[31] Wikipedia, “Pruebas de integración.” [Online]. Available: http://es.wikipedia.org/wiki/Pruebas_de_integración. [Accessed: 05-Jun-2014].

[32] A. Sánchez, “¿Que es Logcat?,” 2013. [Online]. Available: http://infostatex.blogspot.com.es/2013/07/android-que-es-logcat.html. [Accessed: 07-Jun-2014].

[33] “Depuración en Android: Logging,” 2011. [Online]. Available: http://www.sgoliver.net/blog/?p=1916. [Accessed: 07-Jun-2014].

[34] Wikipedia, “Merkle tree.” [Online]. Available: http://en.wikipedia.org/wiki/Merkle_tree. [Accessed: 22-Jun-2014].

[35] Wikipedia, “Criptografía de curva elíptica.” [Online]. Available: http://es.wikipedia.org/wiki/Criptografía_de_curva_elíptica. [Accessed: 29-Jun-2014].

[36] Information Systems Audit and Control Association (ISACA), “Auditoría de Sistemas.” [Online]. Available: http://www.isaca.org/Blogs/282270/archive/2011/04/27/ProteccióndeActivosdeInformación.aspx. [Accessed: 22-Jun-2014].

[37] Boxcryptor, “Cifrado AES y RSA para la nube.” [Online]. Available: https://www.boxcryptor.com/es/cifrado. [Accessed: 04-Jun-2014].

[38] Softpedia, “Android Developer Tools.” [Online]. Available: http://www.softpedia.es/programa-Android-Development-Tools-142527.html. [Accessed: 22-Jun-2014].

[39] Wikipedia, “Interfaz de programación de aplicaciones.” [Online]. Available: http://es.wikipedia.org/wiki/Interfaz_de_programación_de_aplicaciones. [Accessed: 22-Jun-2014].

[40] Alegsa, “¿Cuál es la definicion de Backup?” [Online]. Available: http://www.alegsa.com.ar/Dic/backup.php. [Accessed: 22-Jun-2014].

[41] Wikipedia, “Firmware.” [Online]. Available: http://es.wikipedia.org/wiki/Firmware. [Accessed: 22-Jun-2014].

[42] Wikipedia, “Log (registro).” [Online]. Available: http://es.wikipedia.org/wiki/Log_(registro). [Accessed: 22-Jun-2014].

[43] Wikipedia, “Instituto Nacional de Estándares y Tecnología.” [Online]. Available: http://es.wikipedia.org/wiki/Instituto_Nacional_de_Estándares_y_Tecnología. [Accessed: 22-Jun-2014].

[44] Wikipedia, “Optimal asymmetric encryption padding.” [Online]. Available: http://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding. [Accessed: 04-Jun-2014].

[45] Wikipedia, “Código abierto.” [Online]. Available: http://es.wikipedia.org/wiki/Código_abierto. [Accessed: 22-Jun-2014].

[46] Wikipedia, “PKCS.” [Online]. Available: http://es.wikipedia.org/wiki/PKCS. [Accessed: 22-Jun-2014].

[47] Definición.de, “Definición de plugin.” [Online]. Available: http://definicion.de/plugin/. [Accessed: 22-Jun-2014].

[48] Wikipedia, “RSA.” [Online]. Available: http://es.wikipedia.org/wiki/RSA. [Accessed: 03-Jun-2014].

[49] Wikipedia, “Teléfono inteligente.” [Online]. Available: http://es.wikipedia.org/wiki/Teléfono_inteligente. [Accessed: 22-Jun-2014].

[50] Wikipedia, “Software.” [Online]. Available: http://es.wikipedia.org/wiki/Software. [Accessed: 22-Jun-2014].

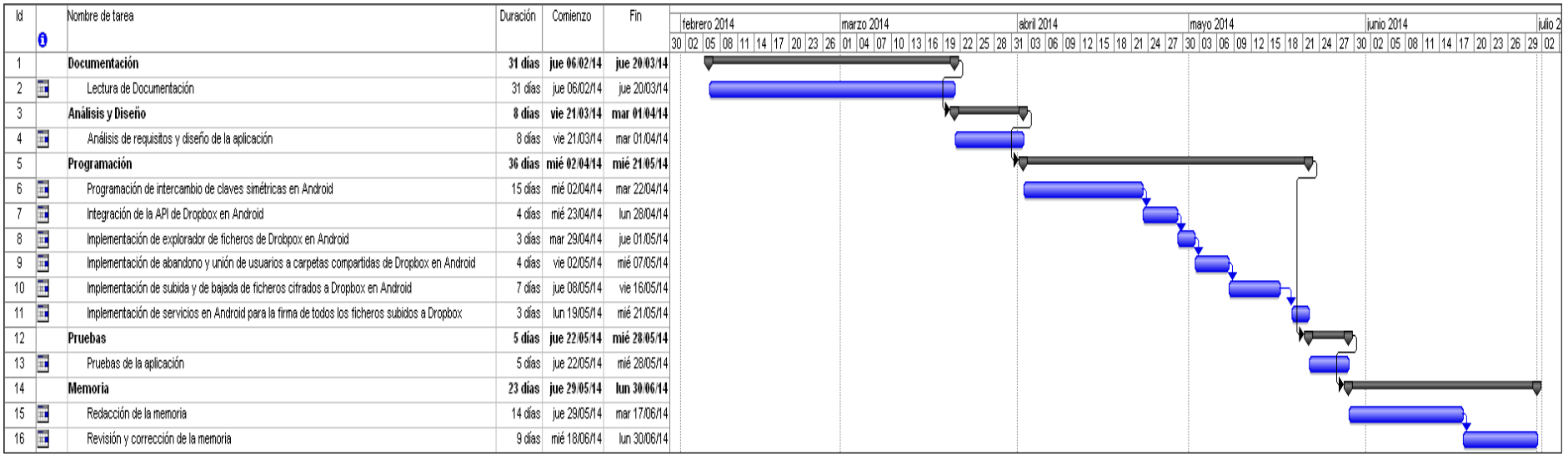
[51] “Definición de TIC.” [Online]. Available: http://www.tics.org.ar/home/index.php/noticias-destacadas-2/157-definicion-de-tics. [Accessed: 28-Jun-2014].

[52] Wikipedia, “Advanced Encryption Standard.” [Online]. Available: http://es.wikipedia.org/wiki/Advanced_Encryption_Standard. [Accessed: 04-Jun-2014].

[53] Wikipedia, “Avalanche effect.” [Online]. Available: http://en.wikipedia.org/wiki/Avalanche_effect. [Accessed: 03-Jun-2014].

[54] F. L. Hernandez, “Criptografía y comercio electrónico con Java,” 2007. [Online]. Available: http://tel.unir.net/~flopez/macprog/sccej.pdf. [Accessed: 23-Jun-2014].

Anexo A. Diagrama de Gantt



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Documentación**	**31 días**	**jue 06/02/14**	**jue 20/03/14**
2	Lectura de Documentación	31 días	jue 06/02/14	jue 20/03/14
3	**Análisis y Diseño**	**8 días**	**vie 21/03/14**	**mar 01/04/14**
4	Análisis de requisitos y diseño de la aplicación	8 días	vie 21/03/14	mar 01/04/14
5	**Programación**	**36 días**	**mié 02/04/14**	**mié 21/05/14**
6	Programación de intercambio de claves simétricas en Android	15 días	mié 02/04/14	mar 22/04/14
7	Integración de la API de Dropbox en Android	4 días	mié 23/04/14	lun 28/04/14
8	Implementación de explorador de ficheros de Drobpox en Android	3 días	mar 29/04/14	jue 01/05/14
9	Implementación de abandono y unión de usuarios a carpetas compartidas de Dropbox en Android	4 días	vie 02/05/14	mié 07/05/14
10	Implementación de subida y de bajada de ficheros cifrados a Dropbox en Android	7 días	jue 08/05/14	vie 16/05/14
11	Implementación de servicios en Android para la firma de todos los ficheros subidos a Dropbox	3 días	lun 19/05/14	mié 21/05/14
12	**Pruebas**	**5 días**	**jue 22/05/14**	**mié 28/05/14**
13	Pruebas de la aplicación	5 días	jue 22/05/14	mié 28/05/14
14	**Memoria**	**23 días**	**jue 29/05/14**	**lun 30/06/14**
15	Redacción de la memoria	14 días	jue 29/05/14	mar 17/06/14
16	Revisión y corrección de la memoria	9 días	mié 18/06/14	lun 30/06/14

Figura 100. Diagrama de Gantt

Anexo B. AES 256, CBC: *Cipher Block Chaining*

El estándar de cifrado avanzado AES, es un algoritmo de clave simétrica, de los más seguros y más utilizados hoy en día. Su historia de éxito comenzó en 1997, cuando el *Instituto Nacional de Estándares y Tecnología* (NIST), anunció la búsqueda de un sucesor para el estándar de cifrado DES. Un algoritmo llamado "*Rijndael*", desarrollado por los criptólogos belgas *Joan Daemen* y *Vincent Rijmen*, fue destacado en seguridad, así como en el rendimiento y la flexibilidad [37].

AES tiene un tamaño de bloque fijo de 128 bits, y en la implementación realizada, la clave es de 256 bits. El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales. Estas operaciones se repiten varias veces, en las llamadas "rondas". El pseudocódigo del AES es como sigue [52]:

- El bloque de texto plano se copia en una matriz denominada «*state*».
- Se realiza una expansión de la clave usando el esquema de claves de *Rijndael*, con el fin de obtener una clave por ronda.
- Etapa inicial:
  1. *AddRoundKey.*
- Rondas:
  1. *SubBytes* — en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.
  2. *ShiftRows* — en este paso se realiza una transposición donde cada fila del «*state*» es rotada de manera cíclica un número determinado de veces.
  3. *MixColumns* — operación de mezclado que opera en las columnas del «*state*», combinando los cuatro bytes en cada columna usando una transformación lineal.
  4. *AddRoundKey* — cada byte del «*state*» es combinado con la clave «*round*». Cada clave «*round*» se deriva de la clave de cifrado usando una iteración de la clave.
- Etapa final:
  1. *SubBytes.*
  2. *ShiftRows.*
  3. *AddRoundKey.*

La robustez del cifrado AES radica en que cumple con los criterios de difusión y confusión, ya que cada bit de la salida depende de todos los bits de la entrada:

$$y_i = f(x_1, x_2, \dots, x_k) \forall i, siendo\ k\ tamaño\ de\ clave\ en\ bits$$

Esto quiere decir que un mínimo cambio en los bits de entrada, supone un cambio total en los bits de salida, tal y como enuncia el efecto de avalancha [53].

La confusión busca ocultar la relación entre el texto cifrado y la clave secreta. En AES, esto se consigue haciendo que los bits de entrada cambien en función de la clave de cada ronda. Esto se produce en la etapa *AddRoundKey* del cifrado.

Por otro lado, la difusión consiste en la eliminación de cualquier relación estadística entre el mensaje original y su texto cifrado. Esto se consigue mediante la combinación de la sustitución y transposición de bits, ya que distribuye la estructura estadística del mensaje sobre la totalidad del mensaje cifrado. Es decir, examinando el texto cifrado no se obtendrá más información sobre el mensaje original que examinando un texto aleatorio.

También es importante destacar que las cajas de sustitución del AES cumplen con los criterios "*Bit Independence Criterion*" (BIC) y "*Strict Avalanche Criterion*" (SAC). El primero de ellos, establece que cualquier par de bits de salida de las cajas de sustitución debe cambiar independientemente cuando un bit de la entrada se complementa:

$$P(y_i, y_j|\bar{x}_k) = P(y_i|\bar{x}_k)P(y_j|\bar{x}_k) \ \forall i \forall j \forall k$$

El criterio SAC establece que un bit de salida de las cajas de sustitución debe cambiar con probabilidad ½ cuando se complementa un bit de la entrada:

$$P(y_j = 1|\bar{x}_i) = P(y_j = 0|\bar{x}_i) = \frac{1}{2} \forall i \forall j$$

En cuanto al modo de operación, se ha elegido el modo *Cipher Block Chaining* (CBC), en vez del modo estándar que es el llamado *Electronic Code Book* (ECB), ya que como puede observarse en la *Figura 101*, este último modo puede revelar patrones del texto, permitiendo ataques estadísticos y diferenciales.

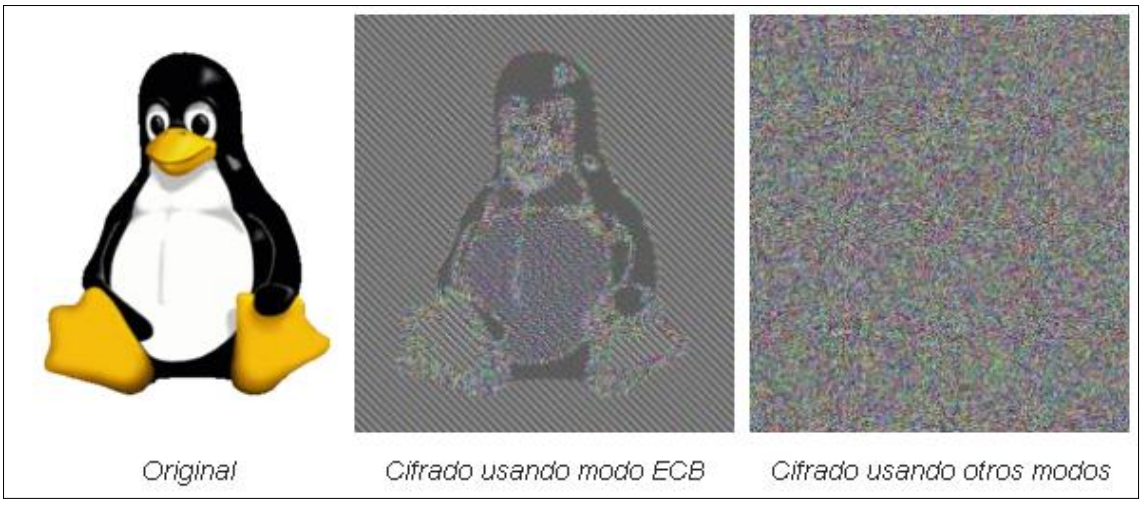


Figura 101. Cifrado ECB

En el modo CBC, antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR con el previo bloque ya cifrado. De esta forma, cada bloque cifrado depende de todos los bloques de texto claros hasta ese punto. Además, para hacer cada mensaje único, se puede usar un vector de inicialización.

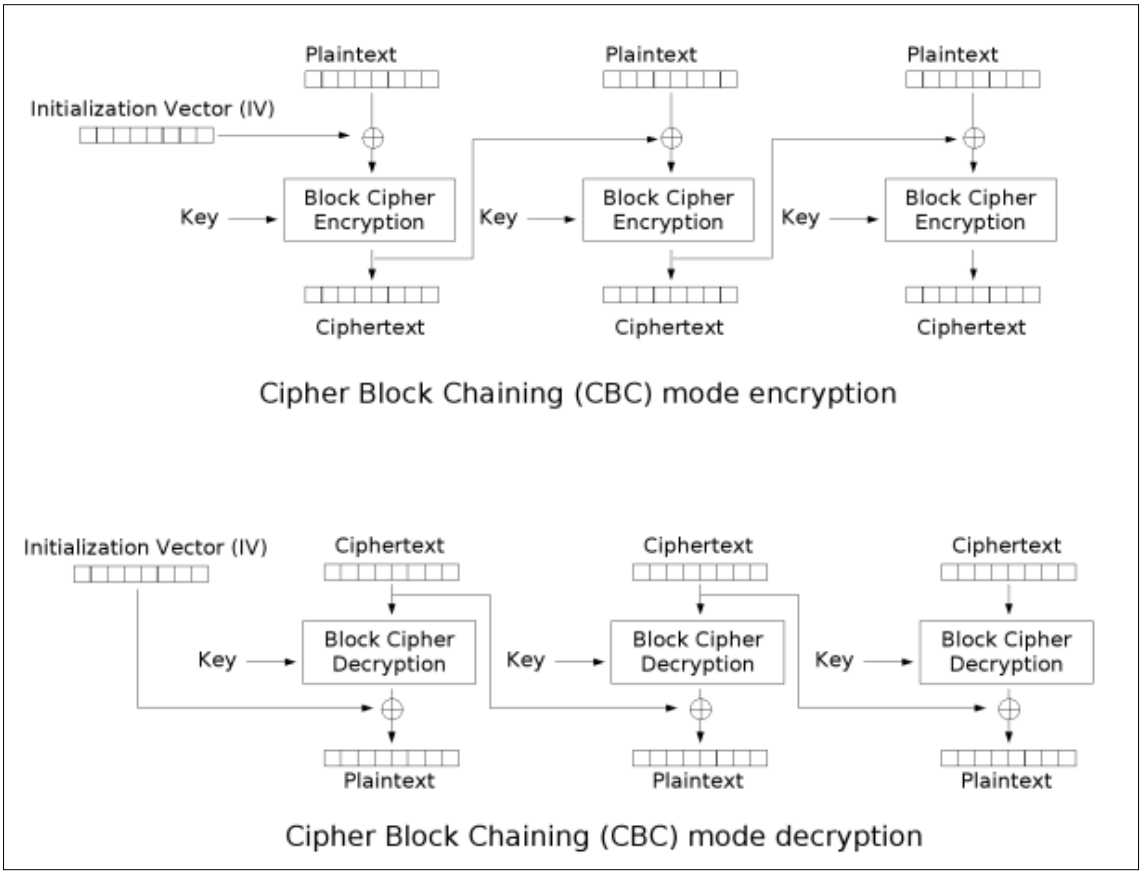


Figura 102. Cifrado y descifrado en modo CBC

B.1. *Padding* en criptografía de clave simétrica

El algoritmo AES 256 trabaja con bloques de datos de 128 bits. Sin embargo, estos datos a cifrar no siempre son múltiplos de este tamaño.

Para resolver este problema hay dos soluciones:

1. Exigir que el usuario pase al algoritmo unidades de información múltiplo del tamaño de bloque.
2. *Padding*. Es decir, que el algoritmo criptográfico rellene el último bloque con datos de relleno antes de cifrarlo, y luego los quite al descifrarlo.

La ventaja de usar *padding* es que es transparente al usuario.

De las técnicas de *padding*, la más utilizada en los algoritmos de claves simétricas es PKCS#5 (*Public Key Cryptography Standard 5*). La técnica consiste en rellenar los bytes restantes del último bloque con un número, que es el número de bytes que han quedado sin rellenar en el último bloque, tal y como muestra la *Figura 103*, para el caso de un tamaño de bloque de 64 bits.

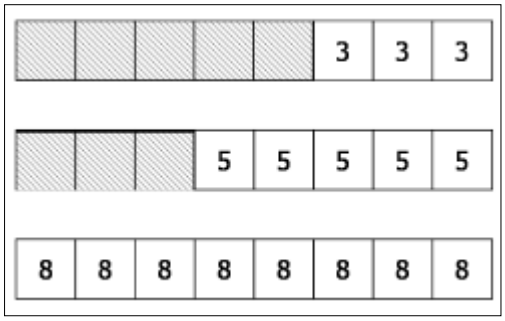


Figura 103. *Padding* PKCS #5

Si resultara que los datos a cifrar fueran múltiplos del tamaño de bloque, se deja un bloque más entero relleno con ochos, en el que caso de que el tamaño de bloque fuese de 64 bits. Esto se hace así para que el receptor pueda detectar el *padding* [54].

Anexo C. *RSA*

En criptografía, RSA es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente [48].

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, éste se ocupa de descifrarlo usando su clave privada.

El algoritmo de generación de claves es el siguiente:

1. Generar dos números primos distintos p y q, cuyo número de bits sea similar.
2. Calcular:
    a. $n = p \cdot q$, *siendo n el módulo para ambas claves, pública y privada*
    b. $\varphi(n) = (p-1) \cdot (q-1)$ *función de Euler*
3. Seleccionar un exponente de cifrado e, tal que:
    a. $mcd(e, \varphi(n)) = 1$, *e y* $\varphi(n)$ *son coprimos*
    b. $1 < e < \varphi(n)$
4. Calcular el exponente de descifrado d:
    a. $d \equiv e^{-1} \, mod \, \varphi(n)$
5. Las claves son:
    a. Clave pública: (n, e)
    b. Clave privada: (n, d)

En cuanto al cifrado y descifrado, si tenemos el mensaje plano m, el cifrado c será:

$$c \equiv m^e \, mod \, n$$

Y para descifrar el mensaje en clave c:

$$m \equiv c^d \, mod \, n$$

C.1. *Padding* en criptografía de clave pública

Como en la criptografía de clave simétrica, en la criptografía de clave pública también se va a utilizar un *padding*, el conocido como OAEP (*Optimal Asymetric Encription Padding*).

OAEP es un esquema de *padding* usado junto el cifrado RSA.

En el siguiente diagrama mostrado en la *Figura 104*, se indica cómo funciona el OAEP [44]:

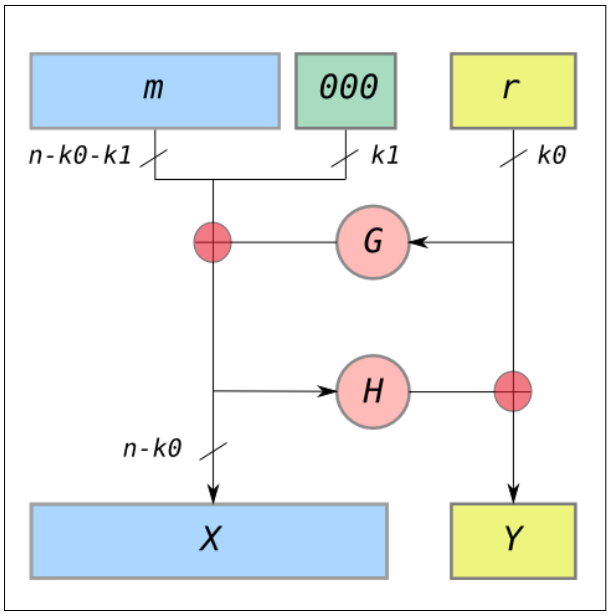


Figura 104. Diagrama OAEP

En el diagrama,

- n es el número de bits del módulo de RSA.
- k_0 y k_1 son dos enteros ajustados por el protocolo.
- m es el texto plano, una cadena de n-k_0-k_1 bits.
- G y H son funciones hash ajustadas por el protocolo.

Para cifrar:

- Al mensaje se le añade un *padding* de k_1 ceros para tener una longitud de n-k_0 bits.
- r es una cadena aleatoria de k_0 bits.
- G expande los k_0 bits de r a n-k_0 bits.
- X = m00...0 XOR G(r)
- H reduce los n-k_0 bits de X a k_0 bits.
- Y = r XOR H(x)
- La salida será la concatenación de X e Y.

Para descifrar:

- Recuperar la cadena aleatoria r = Y XOR H(X)
- Recuperar el mensaje como m00…0 = X XOR G(r)

La seguridad de este esquema reside en que para recuperar m, se tiene que recuperar X e Y, ya que X es requerido para recuperar r a partir de Y, y r es requerido para recuperar m a partir de X. Ya que si se cambia un bit de un *hash* se cambia el resultado completamente, X e Y deben ser recuperados completamente.

Anexo D. Resultados de las pruebas unitarias realizadas

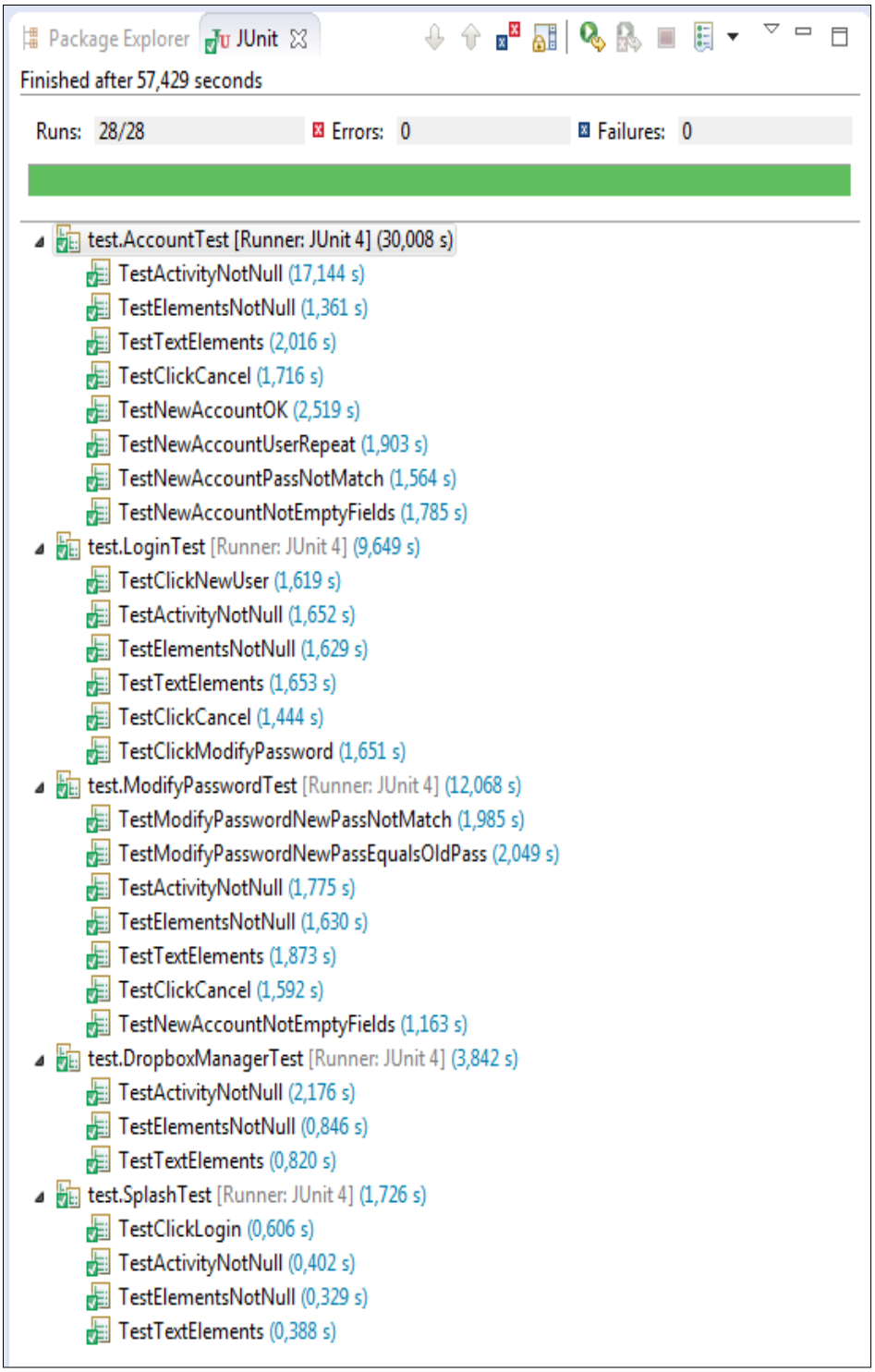


Figura 105. Resultados de las pruebas unitarias realizadas

Anexo E. Resultados de pruebas de integración realizadas

E.1. Importación de una base de datos

En la actividad inicial, llamada ***Splash***, se encuentra el botón con la etiqueta "Importar Base De Datos", encargado de sobrescribir una base de datos local ya existente a partir de otra almacenada en la *sdcard* del dispositivo móvil.

La primera prueba, mostrada en la *Figura 106*, consistió en comprobar que si no hay disponible una base de datos en la *sdcard*, la importación será errónea. Para realizar esta prueba, se elimina si existiera el archivo *encryptedCloud.db*, encontrado en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil. A continuación, se arranca la aplicación y se pulsa sobre el botón "Importar Base De Datos". El resultado esperado de la prueba es que se le muestre un error al usuario.

Figura 106. Resultados prueba importación base de datos (I)

La segunda prueba consiste en realizar una importación correcta. Primero, hay que asegurarse de que hay una base de datos en la *sdcard*. A continuación, se desinstala la aplicación y se vuelve a instalar, para eliminar su base de datos local. Se accede a la pantalla de ***Login*** y se intenta entrar con el usuario "alex" y la contraseña "a" (se trata de una cuenta antigua). Se verifica que se muestra un error. Se vuelve a la pantalla inicial, se pulsa sobre el botón "Importar Base De Datos", la importación se realiza correctamente, y por último se verifica que ahora sí el anterior *login* funciona, como se puede ver en la *Figura 107*.

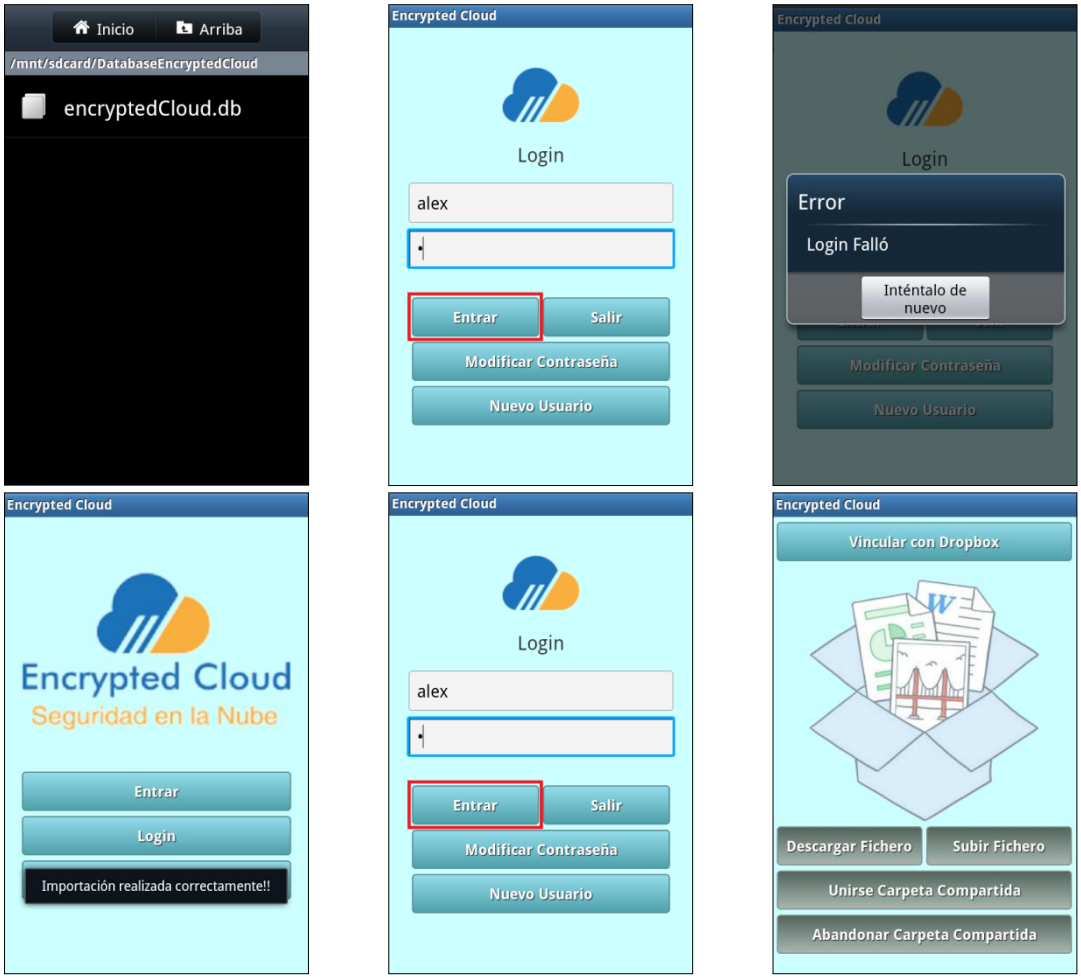


Figura 107. Resultados prueba importación base de datos (II)

E.2. Exportación de una base de datos

Cada vez que se accede a la actividad ***DropboxManager***, se realiza un *backup* de la base de datos de la aplicación en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil. Esto permite que si se desinstala la aplicación, el usuario por ejemplo pueda recuperar sus antiguas credenciales.

La primera prueba realizada, mostrada en la *Figura 108*, comprueba que se copia la base de datos de la aplicación cuando se entra en la actividad ***DropboxManager*** a través de la actividad ***Splash***. Primero se verifica que no existe ningún archivo en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil. A continuación se accede a la aplicación, en la cual ya se habrá hecho un *logueo* previamente, por lo que pulsando sobre el botón "Entrar" de la actividad inicial, se navegará hacia ***DropboxManager***. Por último, se comprobará que se ha generado en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil un archivo llamado *encryptedCloud.db.*

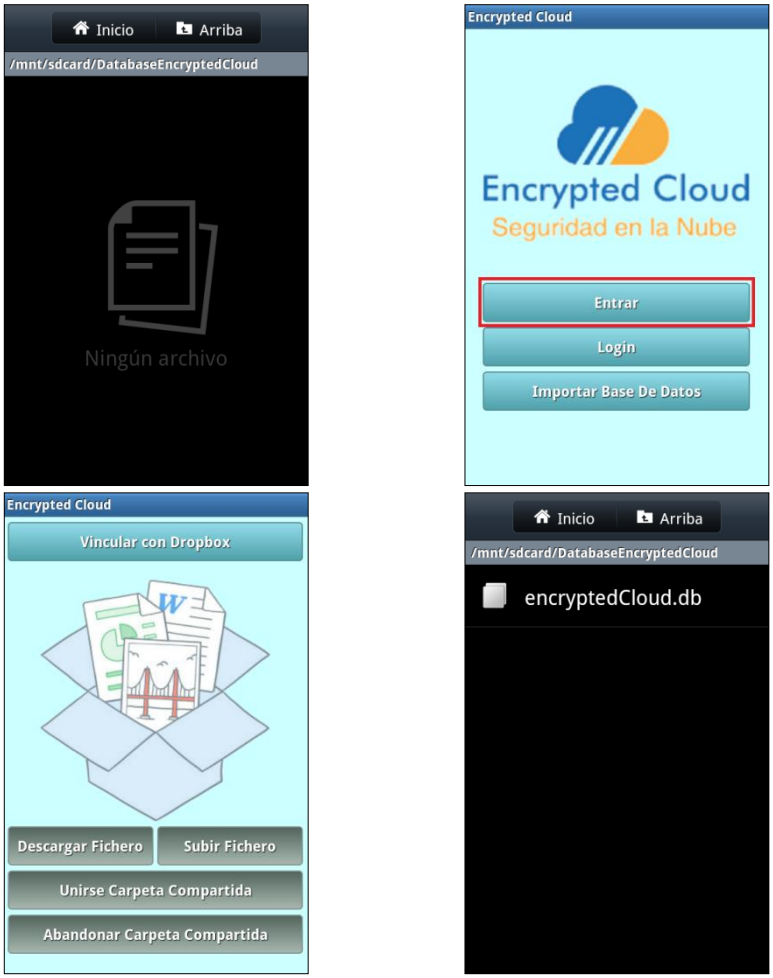


Figura 108. Resultados prueba exportación base de datos (I)

La segunda prueba, que se detalla en la *Figura 109* y en la *Figura 110*, consiste en comprobar que se realiza la exportación de la base de datos accediendo a la actividad ***DropboxManager*** a partir de la actividad de ***Login***. Para ello, simplemente habrá que *loguearse* con una cuenta de usuario previamente creada, y comprobar que se ha generado en la ruta */mnt/sdcard/DatabaseEncryptedCloud* del dispositivo móvil un archivo llamado *encryptedCloud.db*.

Figura 109. Resultados prueba exportación base de datos (II)

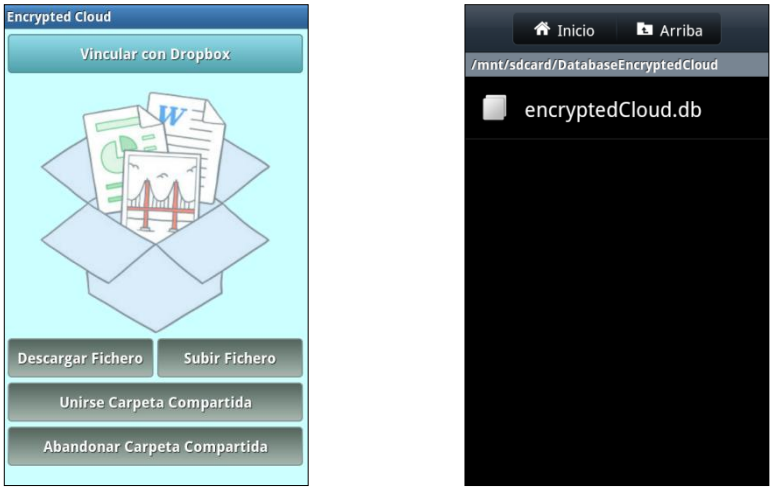


Figura 110. Resultados prueba exportación base de datos (III)

E.3. *Login* en la aplicación

Para poder acceder a la actividad principal de la aplicación, es imprescindible *loguearse* con una cuenta de usuario previamente creada.

La primera prueba realizada consiste simplemente en realizar un *logueo* con una cuenta de usuario inexistente. Se comprueba que se muestra un error al usuario, y se le impide el acceso a la actividad principal ***DropboxManager***, tal y como se puede ver en la *Figura 111*.

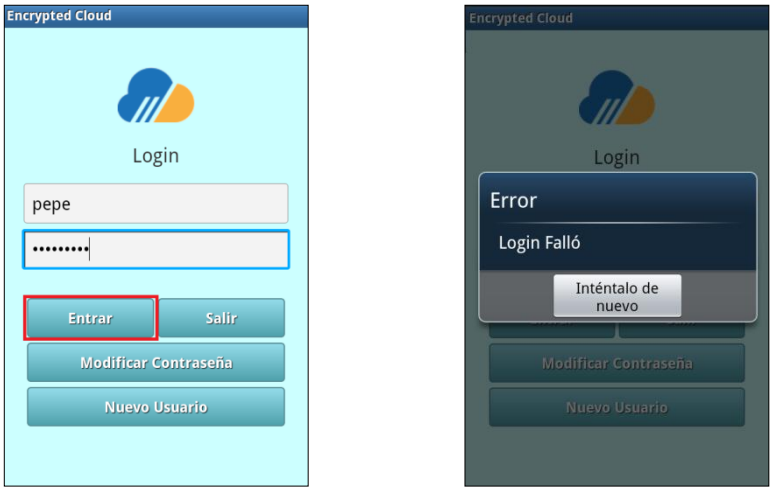


Figura 111. Resultados prueba *Login* (I)

La segunda prueba realizada, ilustrada en la *Figura 112*, es la opuesta a la anterior, es decir, realizar un *logueo* introduciendo unas credenciales correctas. Se comprueba que se le permite al usuario acceder a la actividad principal ***DropboxManager***.

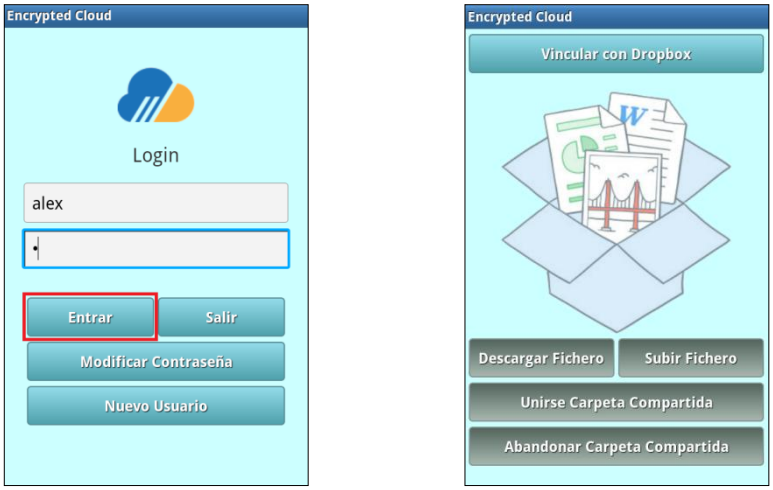


Figura 112. Resultados prueba *Login* (II)

E.4. Modificación de la contraseña de una cuenta de usuario

La aplicación implementada ofrece la funcionalidad al usuario de poder modificar la contraseña de una cuenta de usuario previamente creada.

Primero se probó que cuando se intenta modificar la contraseña de una cuenta inexistente, se informa del error al usuario, como se puede ver en la *Figura 113*.

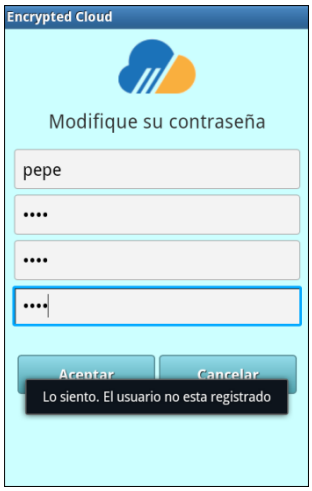


Figura 113. Resultados prueba modificación de contraseña (I)

Por otro lado, se comprobó que cuando se modifica la contraseña de una cuenta previamente creada, realmente se realiza esa modificación. Para probarlo, simplemente se tuvo que hacer un *logueo* con las nuevas credenciales, comprobando que no hay ningún problema en acceder a la actividad principal, ***DropboxManager***. Este proceso es detallado en la *Figura 114*.

Figura 114. Resultados prueba modificación de contraseña (II)

E.5. Navegación a la actividad *DropboxManager* a través de la actividad inicial

A través de la actividad ***Splash***, se puede ir hacia la actividad ***DropboxManager*** pulsando sobre el botón "Entrar", siempre y cuando se haya hecho previamente al menos un *logueo* en la aplicación.

La primera prueba realizada consiste en pulsar el botón "Entrar" sin previamente haber hecho ningún *logueo*. Para ello, simplemente basta con desinstalar e instalar de nuevo la aplicación, y pulsar sobre el botón "Entrar" que se encuentra en la actividad inicial. En la *Figura 115* se muestra el resultado esperado, ya que aparece un error indicando que el usuario debe *loguearse* previamente.

Figura 115. Resultados prueba Entrar *DropboxManager* (I)

La segunda prueba, detallada en la *Figura 116*, comprobó que si ya se ha realizado un *logueo* previamente, un usuario puede acceder a la actividad ***DropboxManager*** simplemente pulsando sobre el botón "Entrar". Para realizar esta prueba, se *logueó* un usuario, y posteriormente, al pulsar sobre el botón "Entrar" de la actividad inicial, se accedió a la actividad ***DropboxManager*** sin problema.

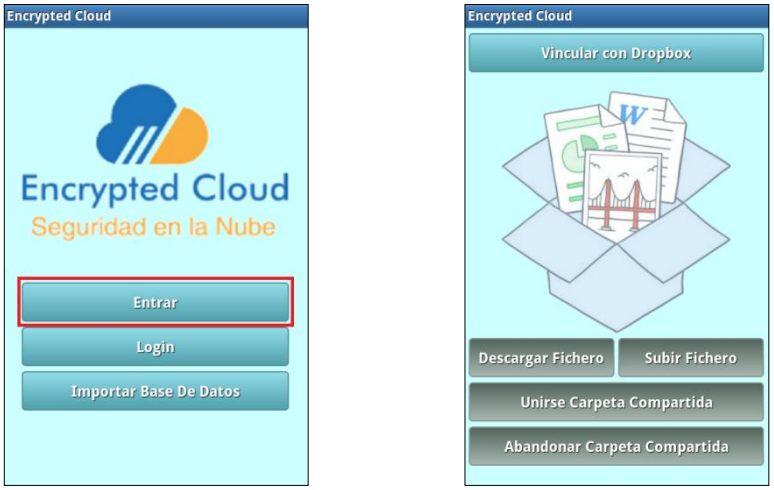


Figura 116. Resultados prueba Entrar *DropboxManager* (II)

E.6. Creación de un nuevo usuario en la aplicación

Otra funcionalidad imprescindible para poder usar la aplicación es la creación de un nuevo usuario. Esta funcionalidad ya fue probada mediante una prueba unitaria, pero en este apartado se quiere mostrar además que el registro se guarda correctamente en la base de datos.

Para llevar a cabo la prueba, simplemente se tuvieron que rellenar los campos usuario y repetir dos veces la contraseña, pulsando a continuación sobre el botón "Aceptar". Tal y como se esperaba, en la *Figura 117* se muestra un mensaje al usuario indicando que se ha creado la cuenta correctamente, y además se verifica que el registro se ha insertado en la base de datos en la tabla ***User*** sin ningún tipo de problema.

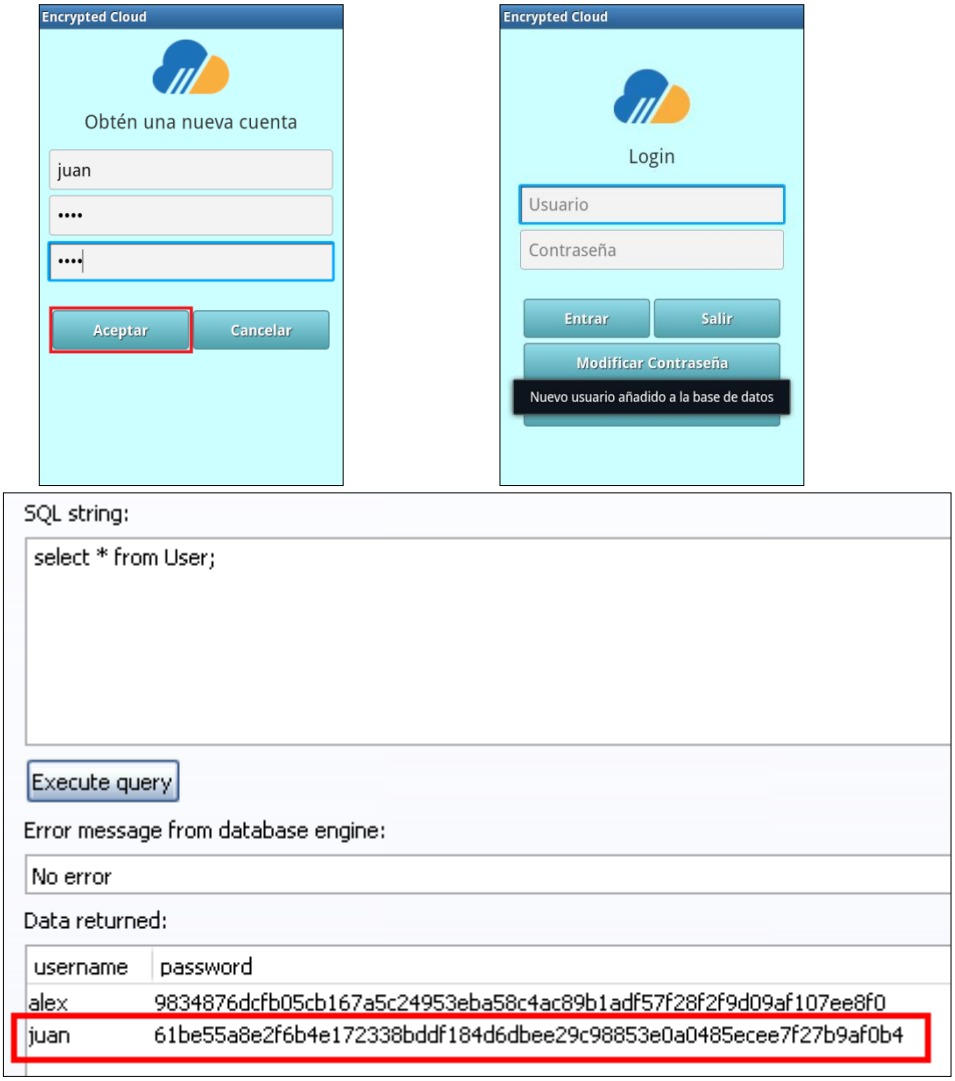


Figura 117. Resultados prueba creación cuenta de usuario

E.7. Vincularse a una cuenta de *Dropbox*

Para poder hacer uso de las funcionalidades relacionadas con *Dropbox* que proporciona la aplicación, es imprescindible tener una cuenta de este tipo de servicio creada para poder vincularla.

En la primera prueba que se muestra en la *Figura 118*, se verificó que cuando se intenta vincular una cuenta de *Dropbox* errónea, se muestra un mensaje de error al usuario. Para probar esto, estando en la actividad ***DropboxManager***, se deberá pulsar el botón "Vincular con *Dropbox*". A continuación, se abrirá el navegador del dispositivo móvil, pidiéndose las credenciales de una cuenta. Si estas credenciales no son correctas, se puede verificar que el usuario es informado debidamente.

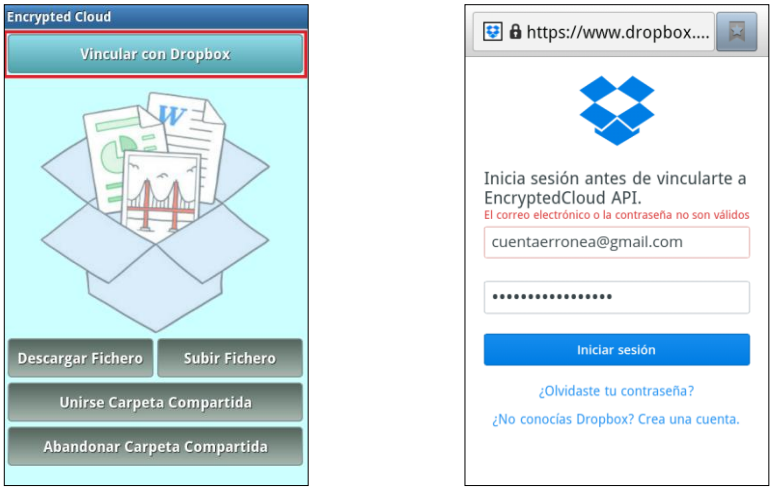


Figura 118. Resultados prueba vinculación cuenta de *Dropbox* (I)

La segunda prueba es la opuesta a la primera, y consiste en que no se encuentre ningún problema al vincular una cuenta existente de *Dropbox* con la aplicación. Para ello, estando en la actividad ***DropboxManager***, se deberá pulsar el botón "Vincular con *Dropbox*". A continuación, se abrirá el navegador del dispositivo móvil, pidiéndose las credenciales de una cuenta. Se rellenarán correctamente y se pulsará sobre el botón "Iniciar sesión". Seguidamente, se navegará a otra pantalla en la cual la aplicación "*EncryptedCloud* API" pedirá permisos para poder acceder a todo el contenido de la cuenta de *Dropbox*. Se pulsará el botón "Permitir", finalizando el proceso de vinculación. Esta prueba se puede ver en la *Figura 119* y en la *Figura 120*.

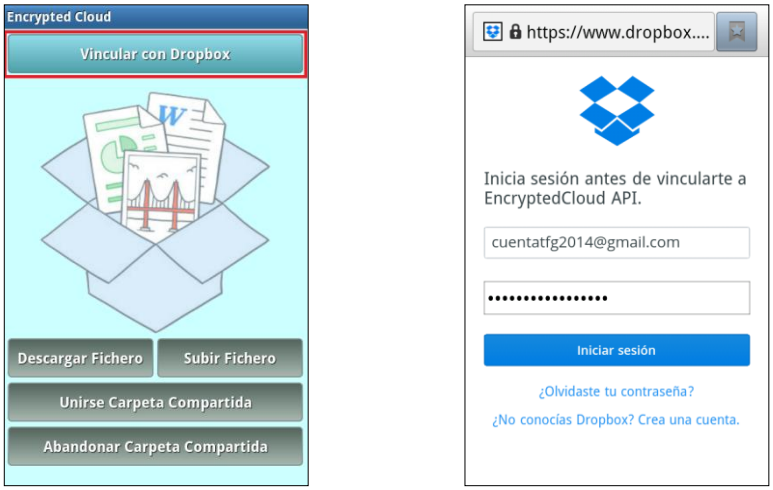


Figura 119. Resultados prueba vinculación cuenta de *Dropbox* (II)

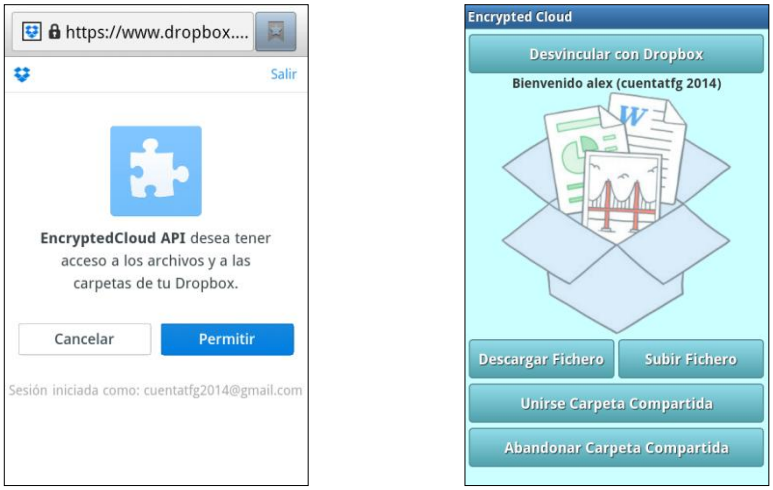


Figura 120. Resultados prueba vinculación cuenta de *Dropbox* (III)

E.8. Desvincularse de una cuenta de *Dropbox*

Si se quiere vincular a una nueva cuenta de *Dropbox* y se está vinculado a otra, se deberá realizar una desvinculación de la cuenta actual.

Esta prueba es muy sencilla, y sólo se quiere tener la seguridad de que la aplicación responde de una manera concreta cuando se realiza una desvinculación. Lo que se comprobó es que, estando en la actividad ***DropboxManager*** vinculado a una cuenta de *Dropbox*, si se pulsa el botón superior "Desvincular con *Dropbox*", automáticamente se bloquean todas las opciones mostradas en la parte inferior de la actividad como se puede comprobar en la *Figura 121*, y que están relacionadas con toda la funcionalidad ofrecida para interactuar con una cuenta de *Dropbox*.

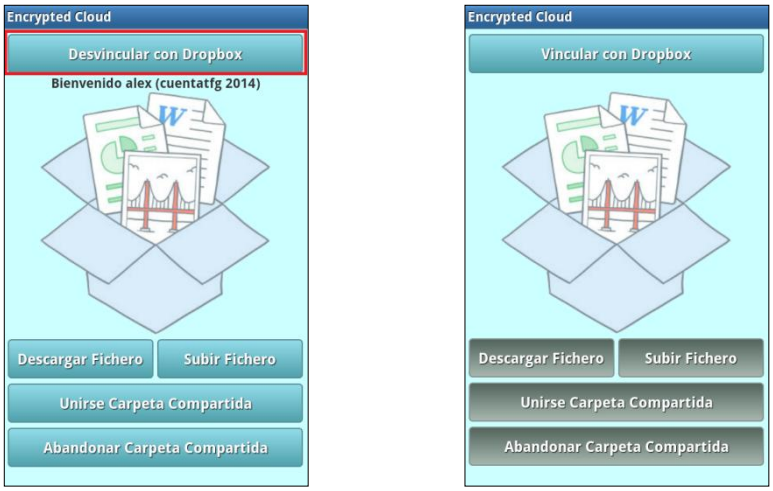


Figura 121. Resultados prueba desvinculación cuenta de *Dropbox* (I)

E.9. Actualización de carpetas compartidas

La aplicación desarrollada tiene un servicio encargado de resolver todas las peticiones de unión y abandono de una carpeta compartida, en caso de que se sea propietario de la misma. En esta sección se va a probar el correcto funcionamiento de la actualización de carpetas compartidas. En estas pruebas se va a trabajar tanto con las peticiones de unión y abandono de una carpeta compartida, como con su fichero *usuariosCarpeta.txt*.

La primera prueba consistió en la gestión por parte del propietario de una carpeta compartida, de dos peticiones de unión solicitadas por otros usuarios. Para ello, primero se revisó el contenido del fichero de *usuariosCarpeta.txt* de la carpeta compartida, verificando que sólo se encontraba el propietario. A continuación, se mandaron dos peticiones de unión a la carpeta compartida, como se muestra en la *Figura 122*.

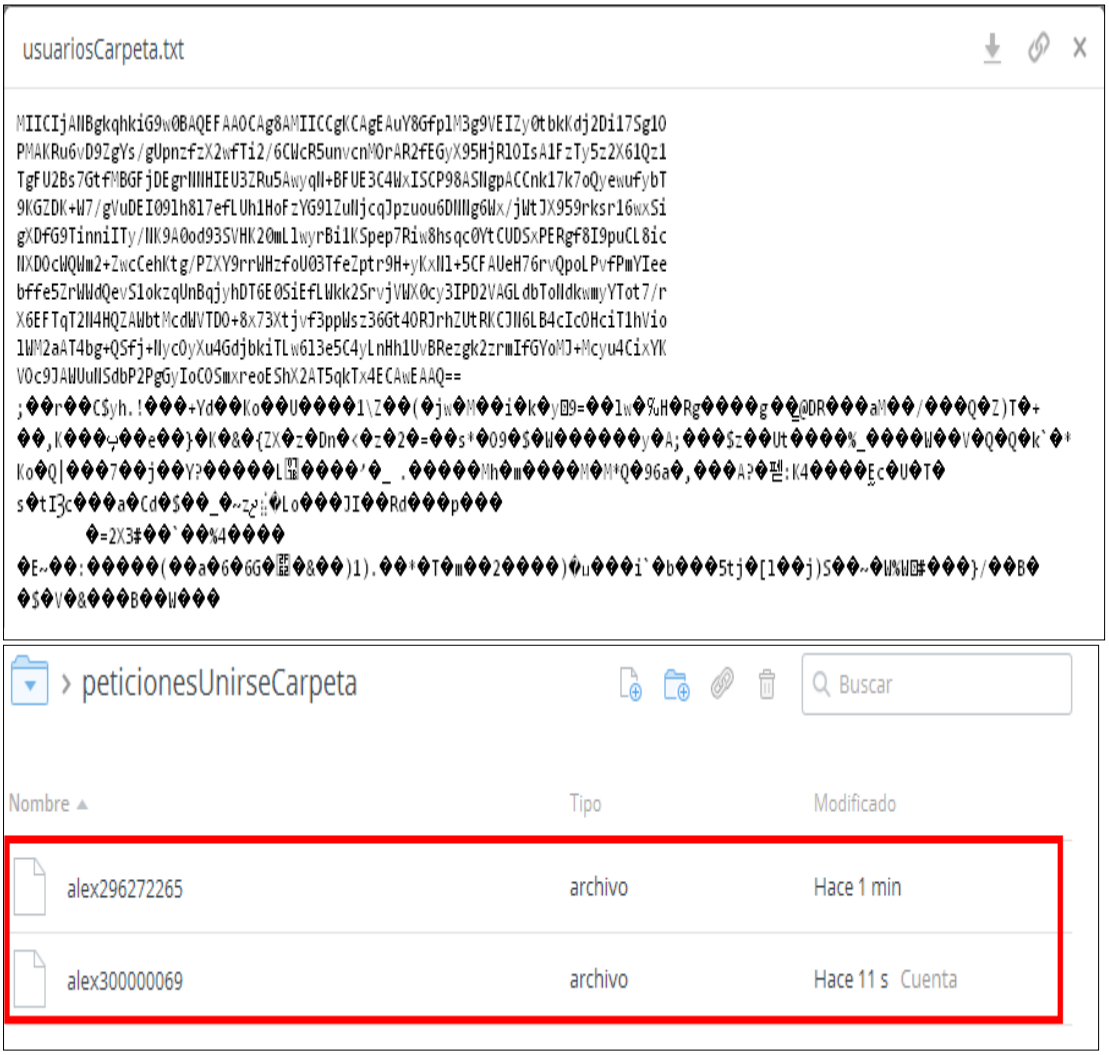


Figura 122. Resultados prueba actualización carpeta compartida (I)

Por último, se verificó en la *Figura 123* que los dos usuarios eran añadidos al fichero *usuariosCarpeta.txt*, y las dos peticiones eran eliminadas.

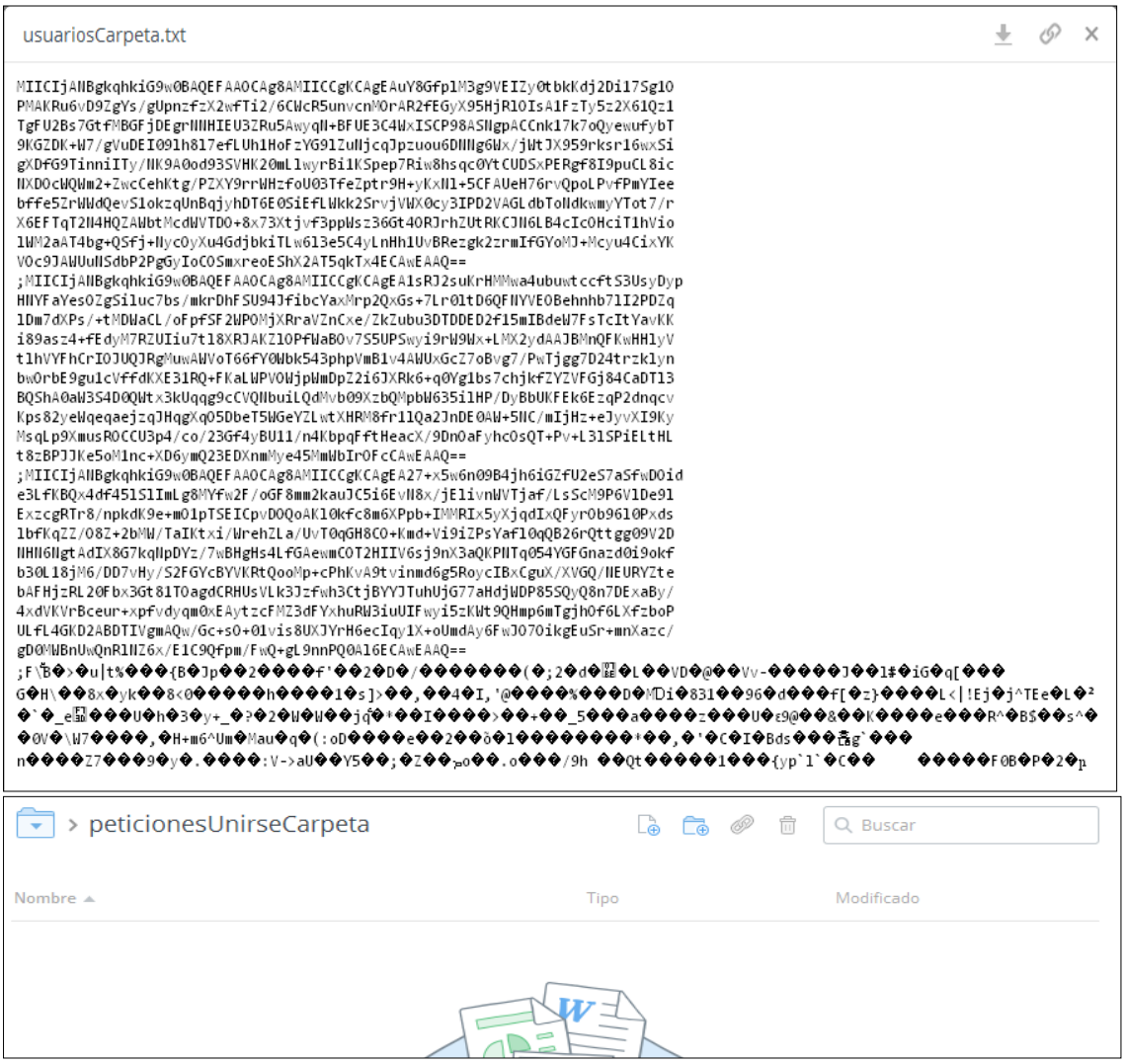


Figura 123. Resultados prueba actualización carpeta compartida (II)

La siguiente prueba fue la misma que la anterior, pero en este caso se gestionaron dos peticiones de abandono. Se partía de una carpeta compartida con tres usuarios, quedándose sólo el propietario cuando se gestionaron las dos peticiones correspondientes. Esta prueba se ha detallado en la *Figura 124* y en la *Figura 125*.

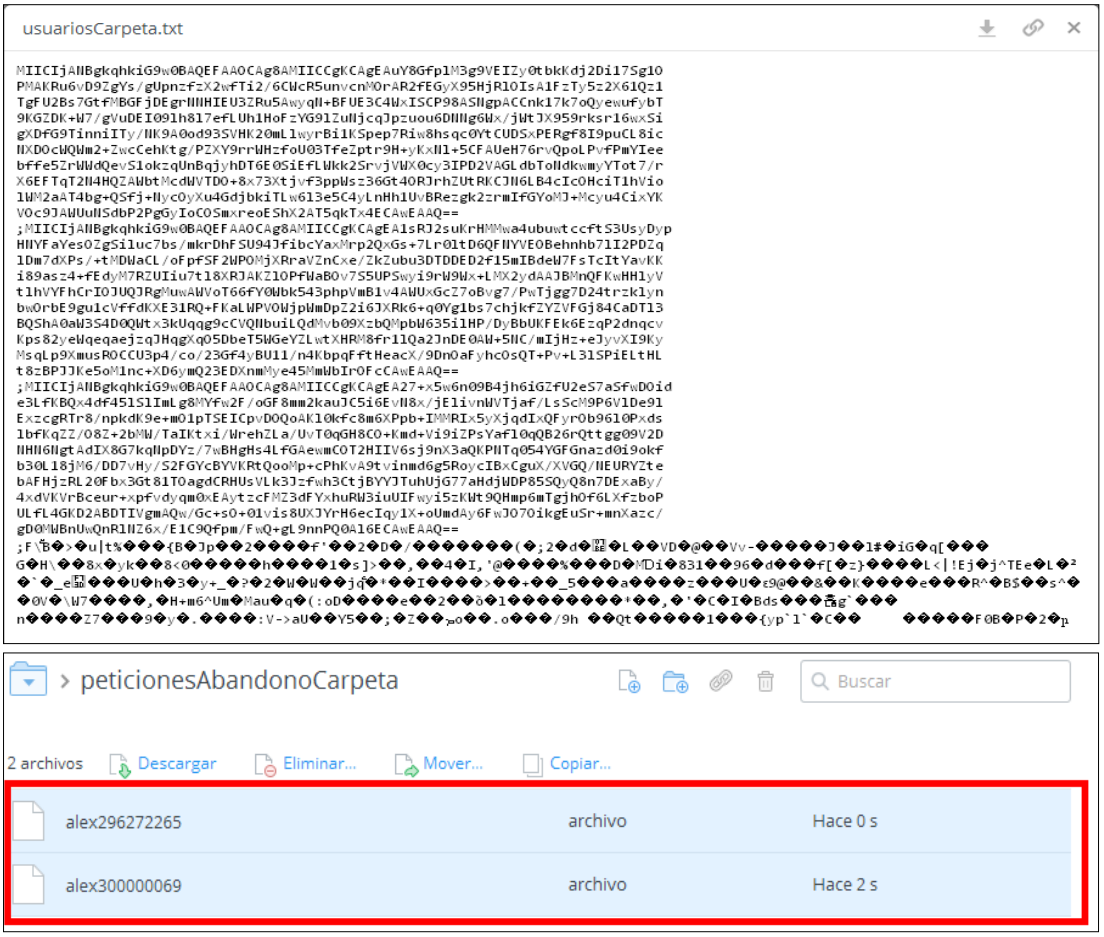


Figura 124. Resultados prueba actualización carpeta compartida (III)

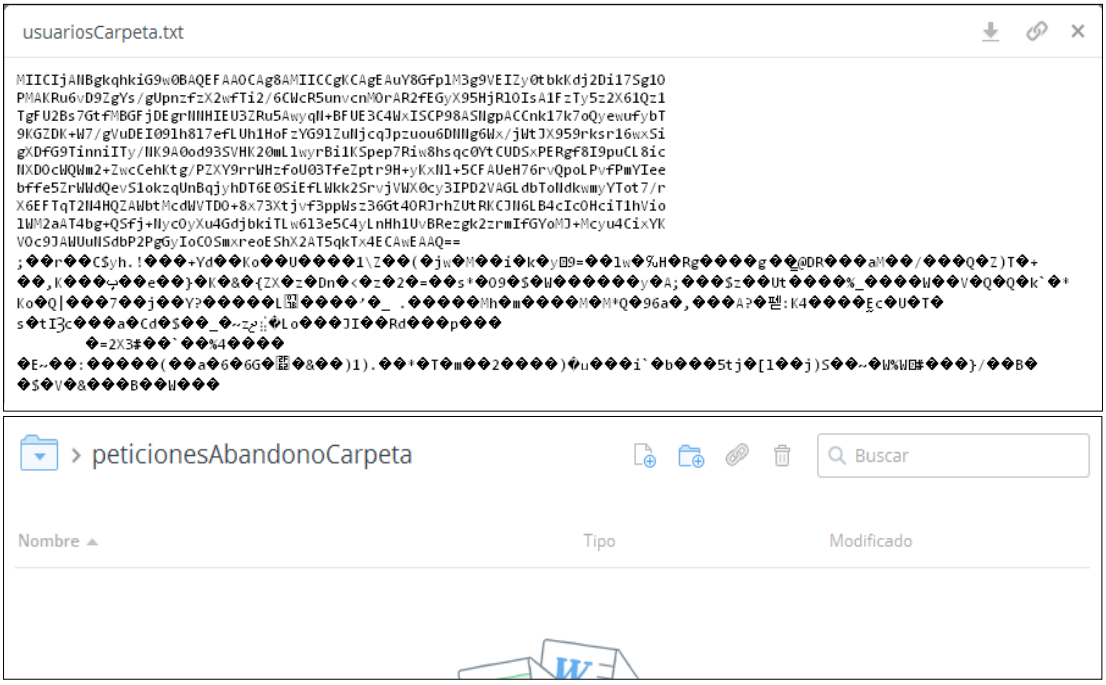


Figura 125. Resultados prueba actualización carpeta compartida (IV)

La tercera prueba consistió en gestionar una petición de unión y una petición de abandono, de una carpeta compartida en la cual se encontraba el propietario y el usuario que quiere abandonar. En la *Figura 126* se pueden ver las dos peticiones realizadas.

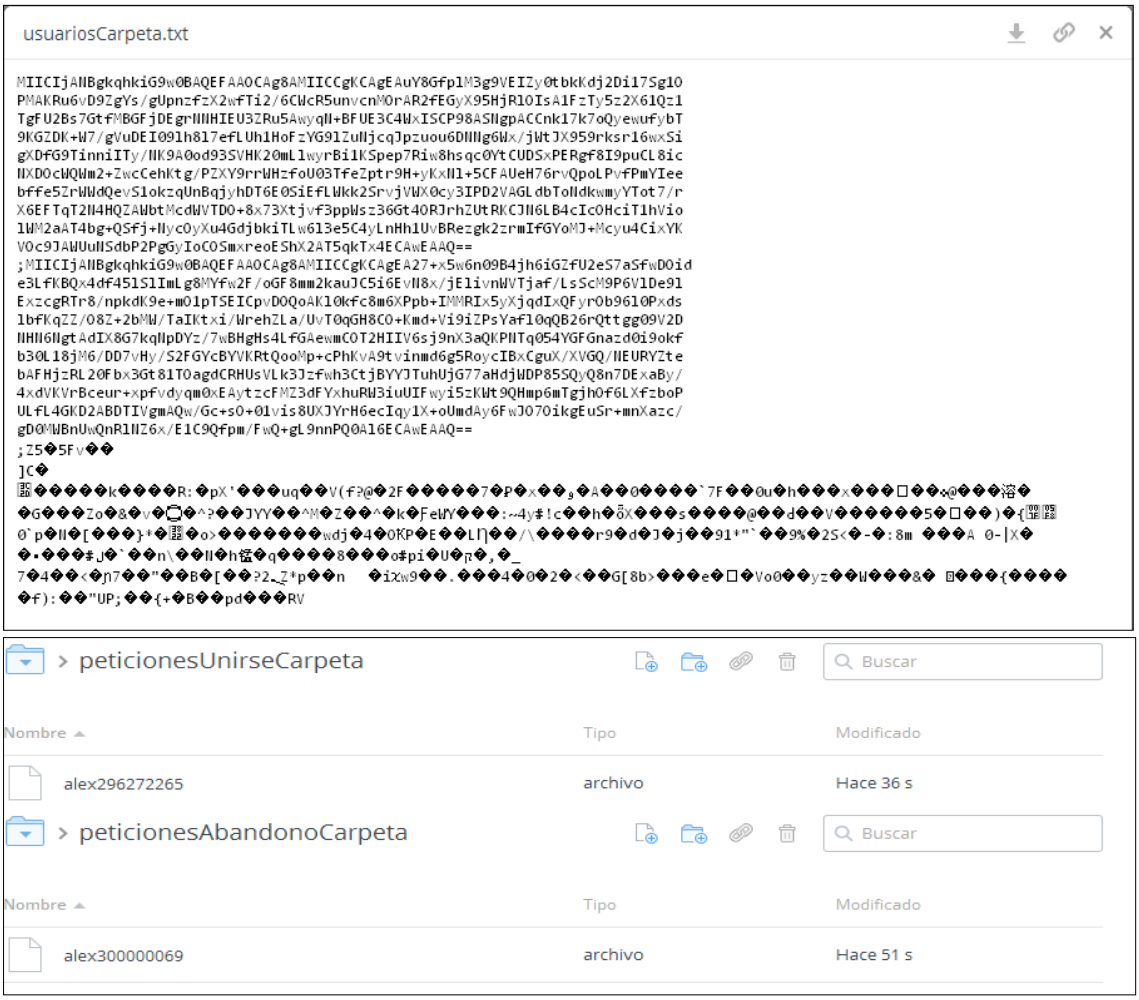


Figura 126. Resultados prueba actualización carpeta compartida (V)

En la *Figura 127*, se comprobó que después de la actualización de la carpeta, se encontraban únicamente el propietario y el usuario que se acababa de unir, además de eliminarse las peticiones mandadas.

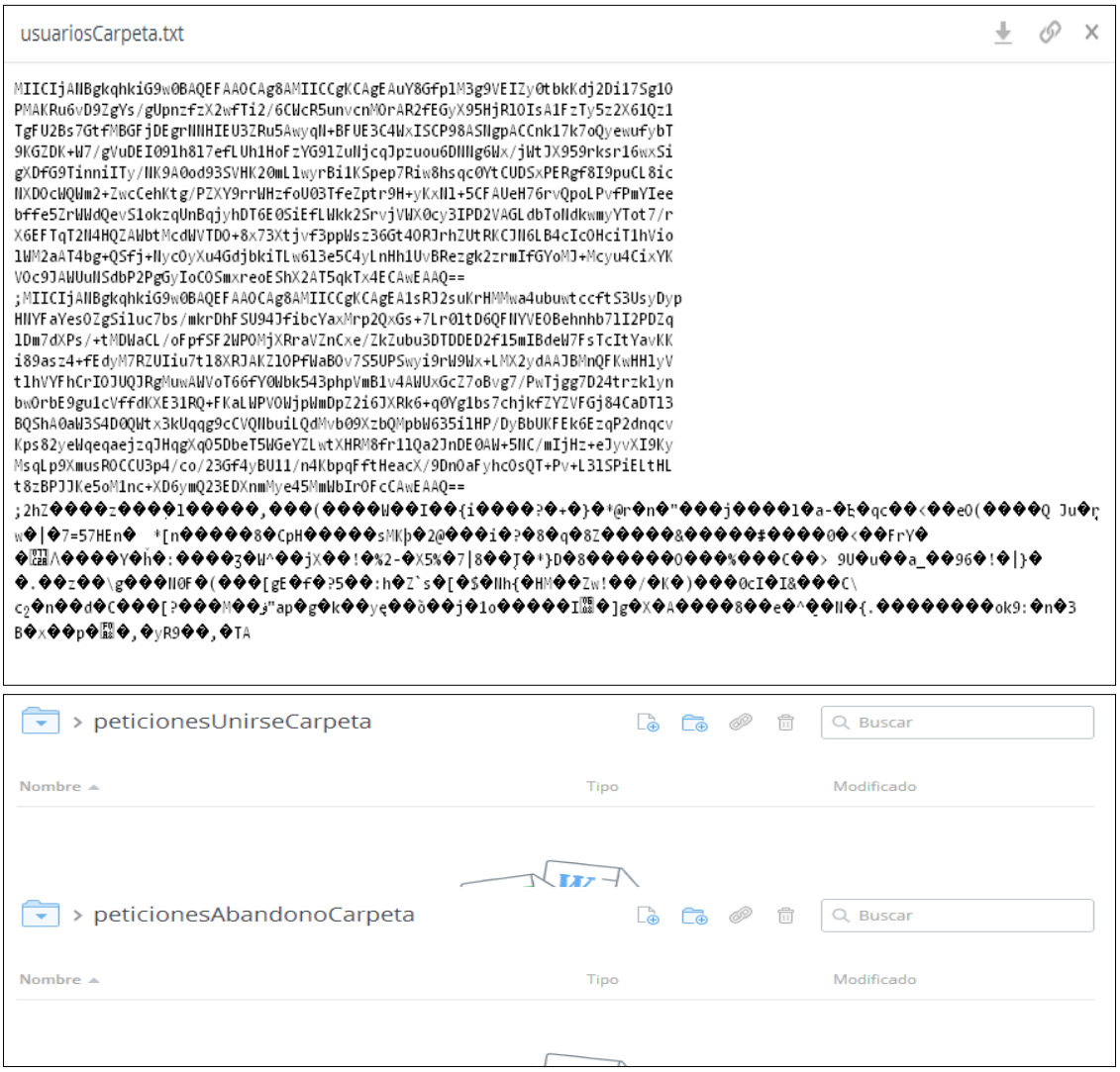


Figura 127. Resultados prueba actualización carpeta compartida (VI)

La cuarta prueba realizada fue para comprobar que si se sube un fichero cualquiera, por ejemplo a la carpeta de peticiones de unión de una carpeta compartida, dicha petición se ignora debido a que no está correctamente firmada. Para comprobarlo, se subió un fichero a la carpeta de peticiones de unión y se verificó que ese fichero es eliminado e ignorado, debido a que los usuarios iniciales son los mismos que los finales después de esa eliminación. Los resultados de esta prueba han sido detallados en la *Figura 128* y en la *Figura 129*.

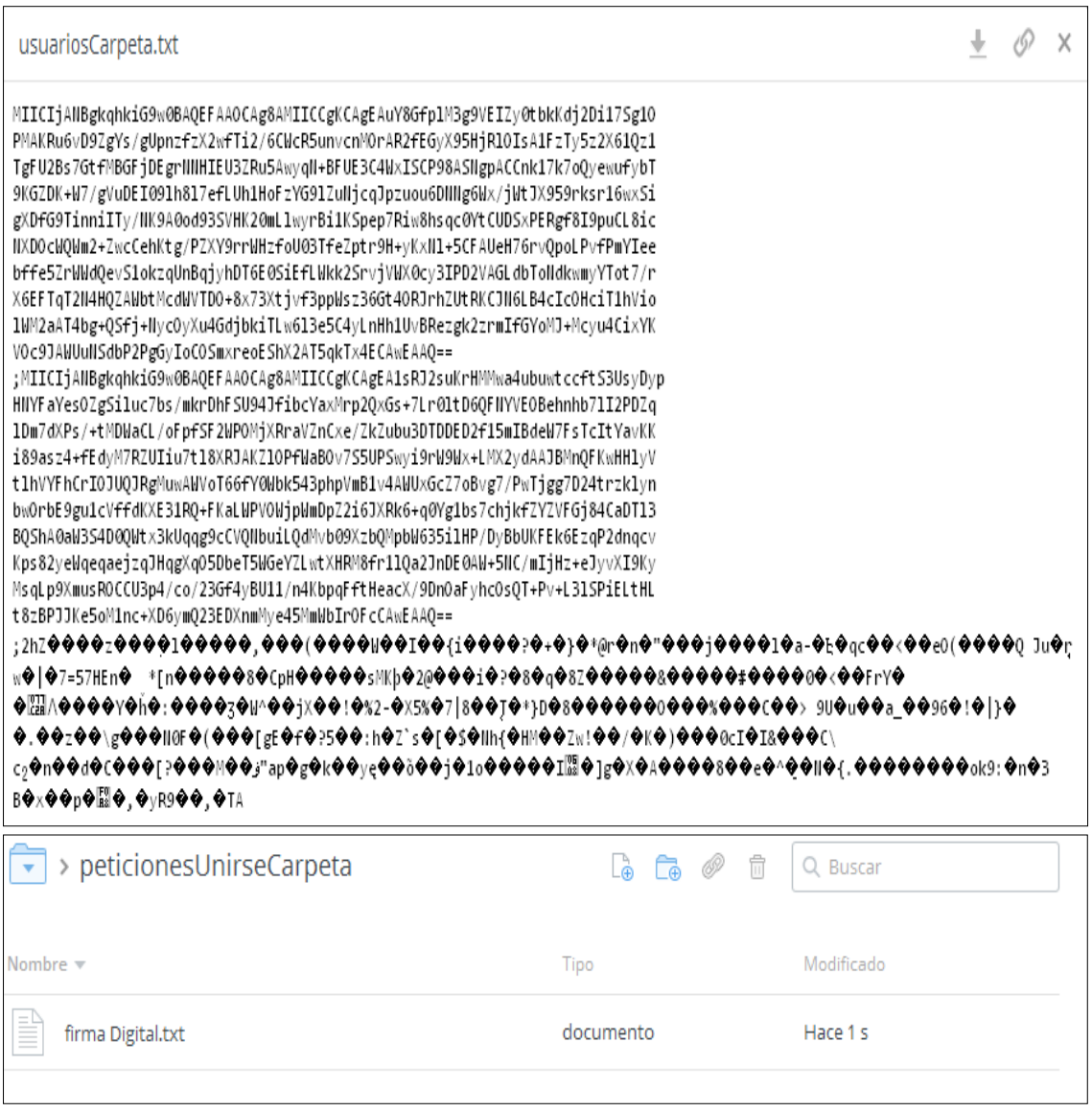


Figura 128. Resultados prueba actualización carpeta compartida (VII)

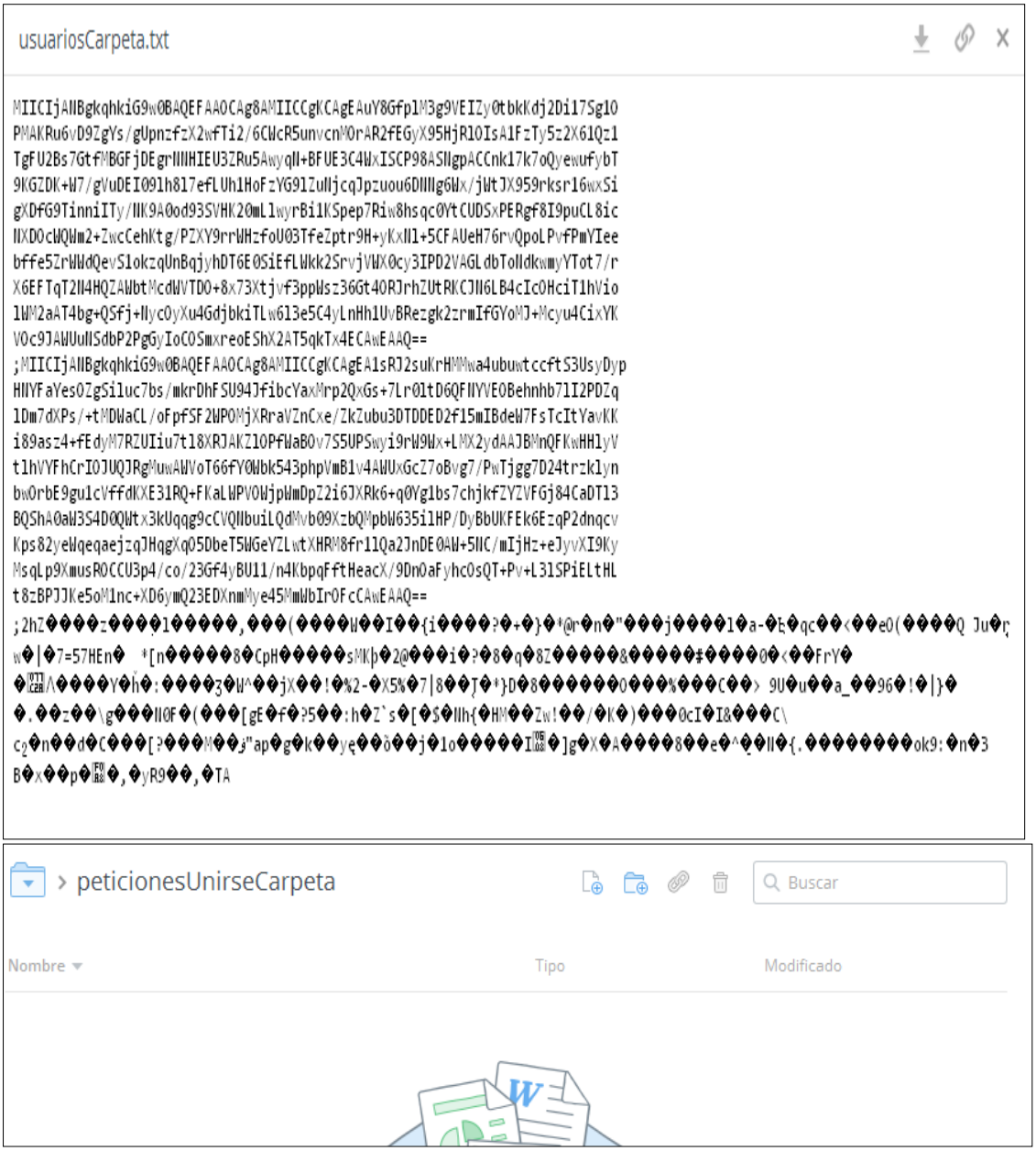


Figura 129. Resultados prueba actualización carpeta compartida (VIII)

La última prueba realizada consistió en modificar el fichero de *usuariosCarpeta.txt*, con el fin de verificar que cuando el propietario de dicha carpeta se dé cuenta de que ese fichero ha sido modificado indebidamente, es capaz de recuperar de la base de datos su versión válida más reciente y de esta forma poder sobrescribir la versión modificada. En la *Figura 130* se puede ver el contenido del fichero de usuarios modificado.

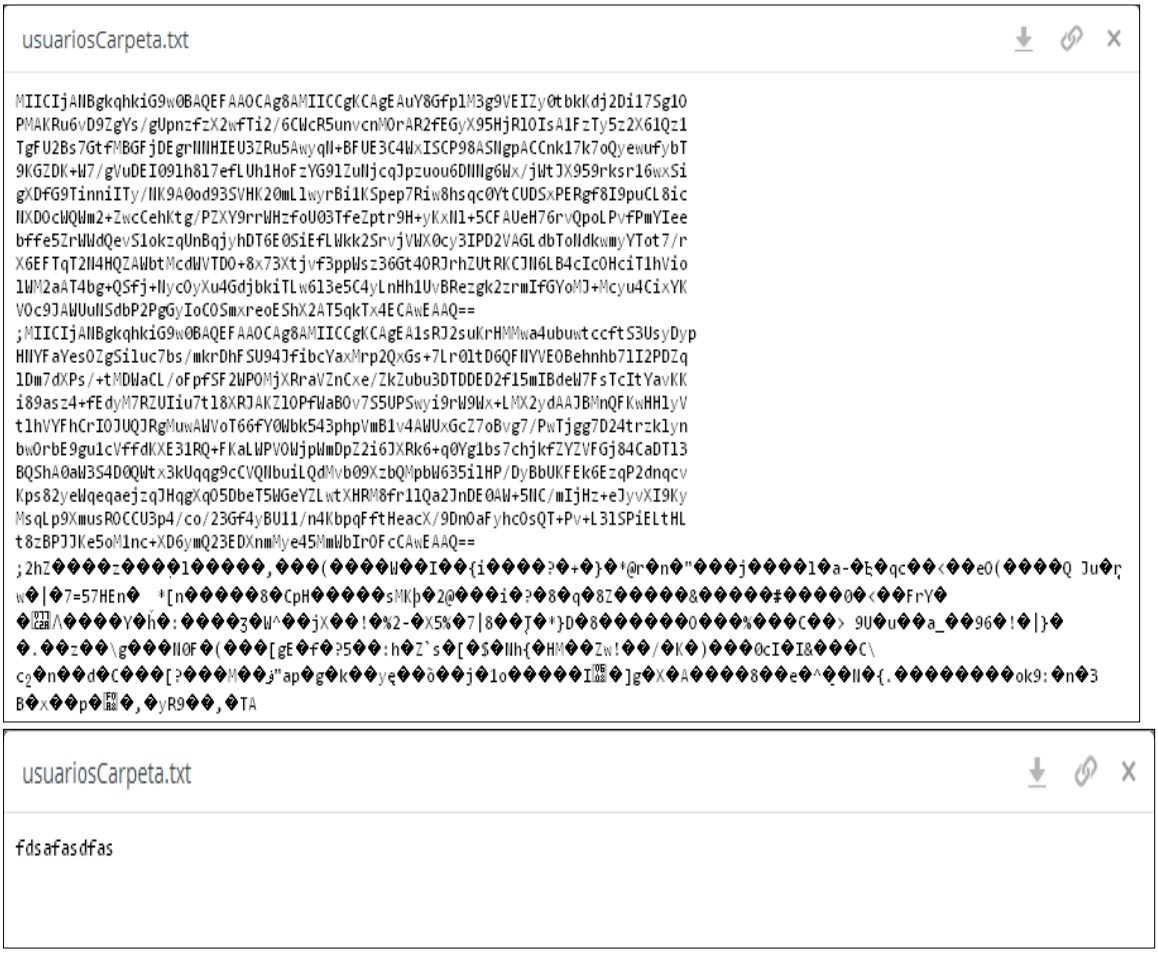


Figura 130. Resultados prueba actualización carpeta compartida (IX)

En la *Figura 131*, se puede comprobar mediante el LogCat, que se realiza un acceso a la base de datos local para poder recuperar el último contenido válido del fichero de usuarios.

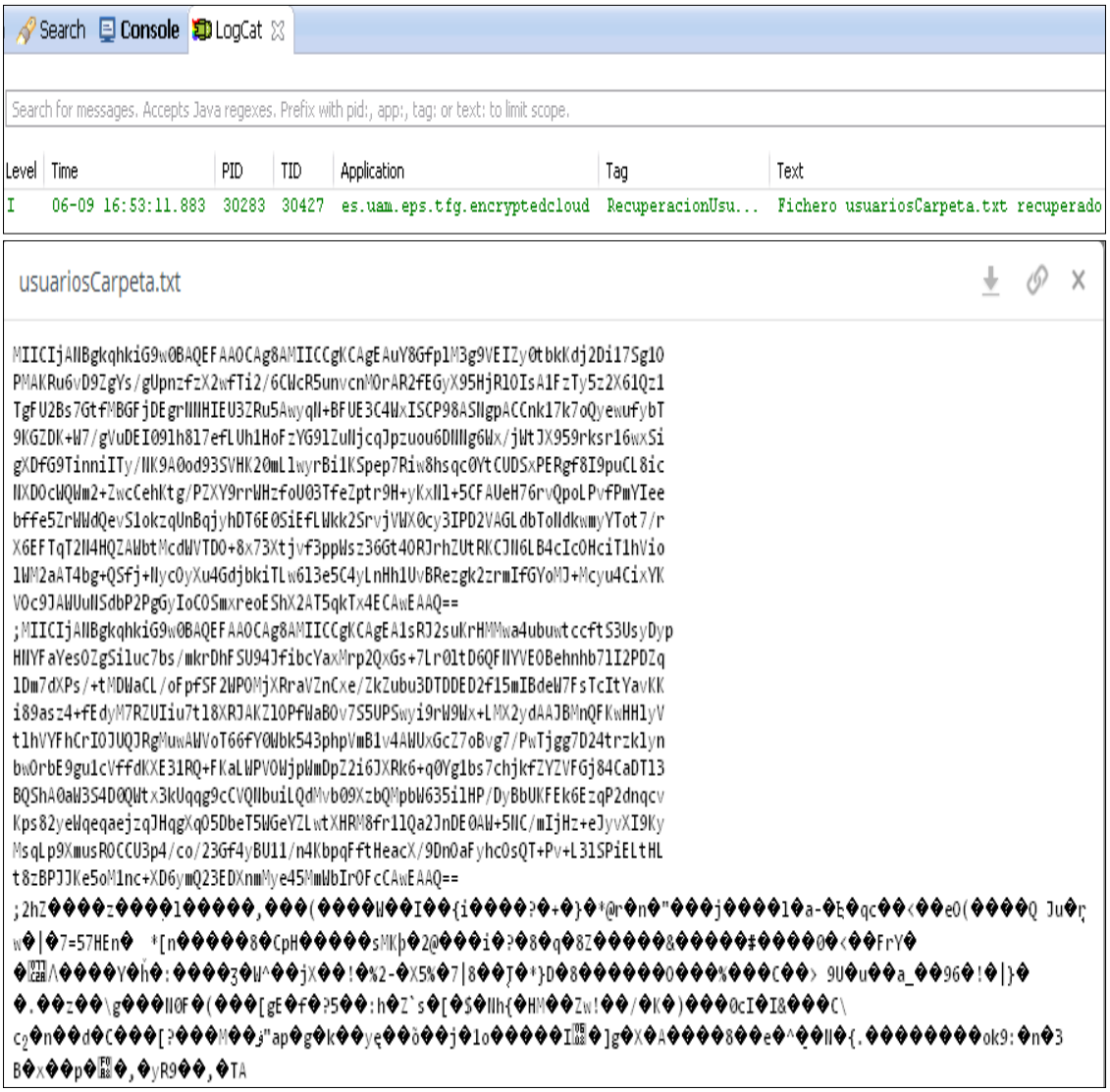


Figura 131. Resultados prueba actualización carpeta compartida (X)

E.10. Eliminación indebida de un fichero de *Dropbox*

Es importante comprobar que cuando se sube un fichero a una cuenta de *Dropbox* vinculada, éste no es eliminado por un tercero.

En la prueba que se va a realizar, se va a subir un fichero a una carpeta no compartida de una cuenta de *Dropbox*, tal y como se muestra en la *Figura 132*.

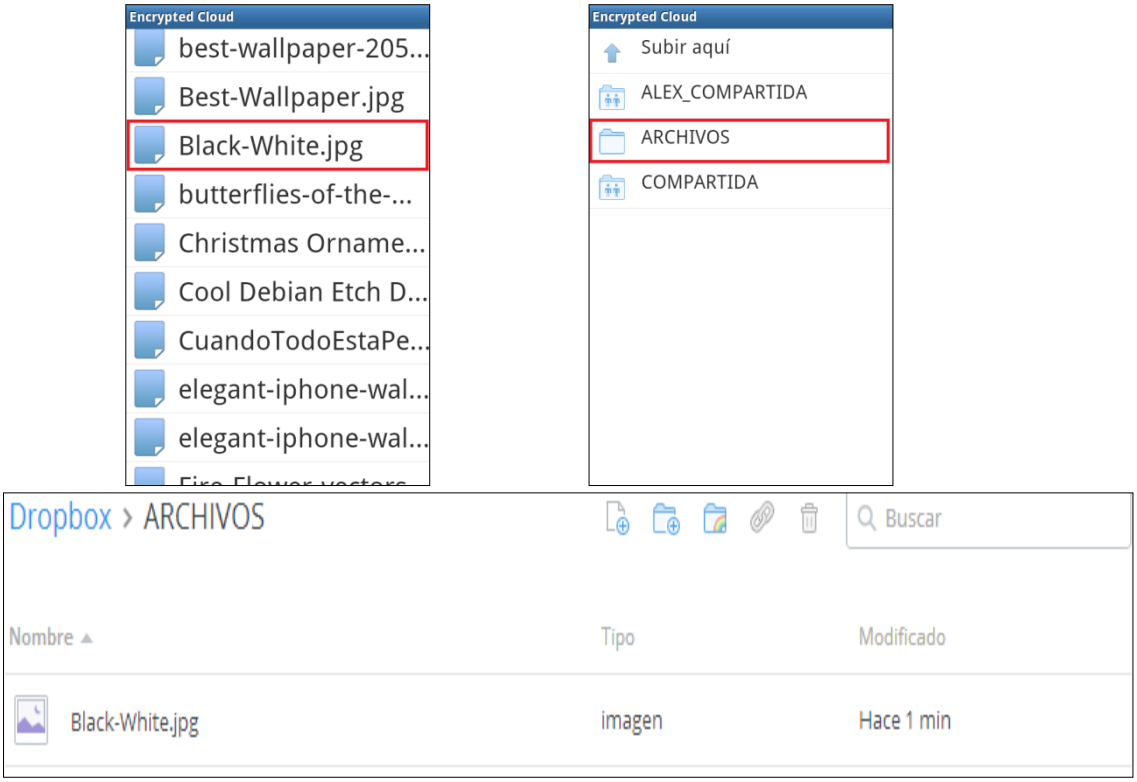


Figura 132. Resultados prueba eliminación indebida de fichero (I)

A continuación, en la *Figura 133* se procede a su eliminación manual para poder comprobar que la aplicación notifica al usuario de que un fichero que ha subido previamente ha sido eliminado. En el caso de subir el fichero a una carpeta compartida se observaría el mismo comportamiento, notificando al propietario de la misma.

Figura 133. Resultados prueba eliminación indebida de fichero (II)